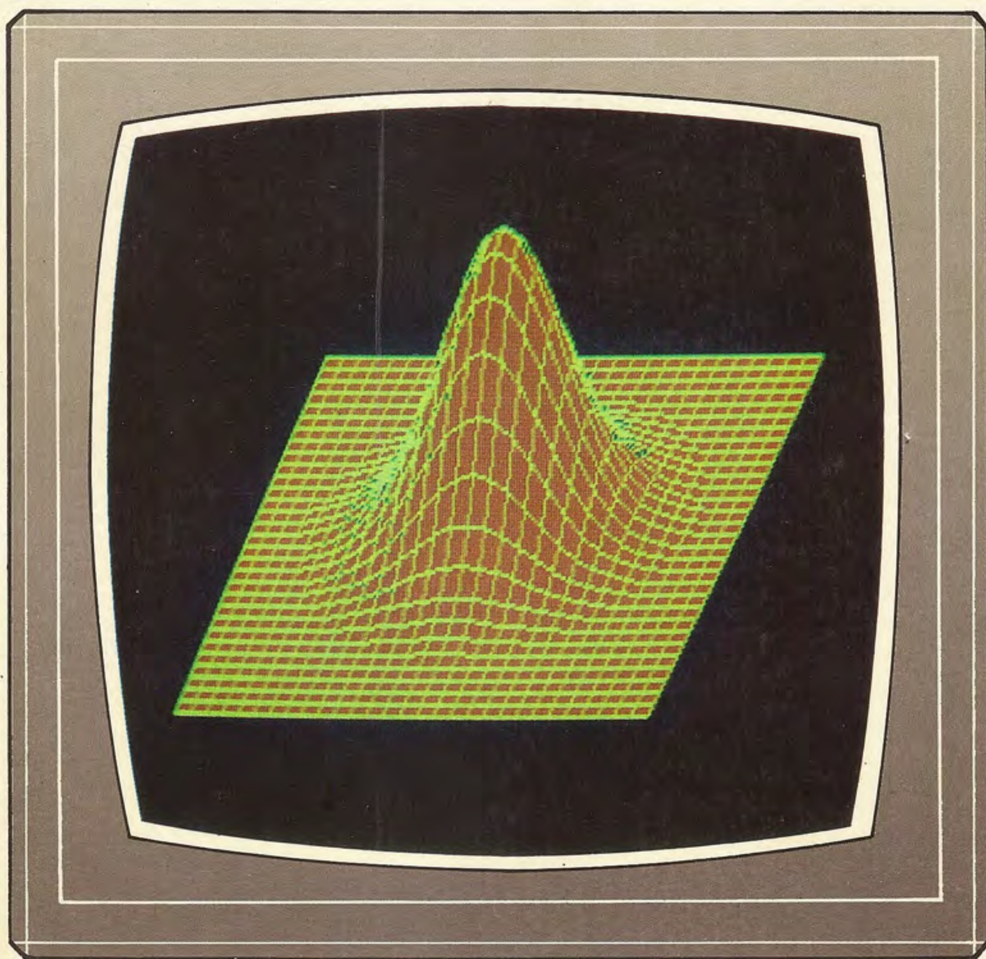
▼ AMSTRAD ▼ SPECTRUM ▼ COMMODORE ▼ IBM ▼ MSX ▼

Informática y programación 1

PASO A PASO



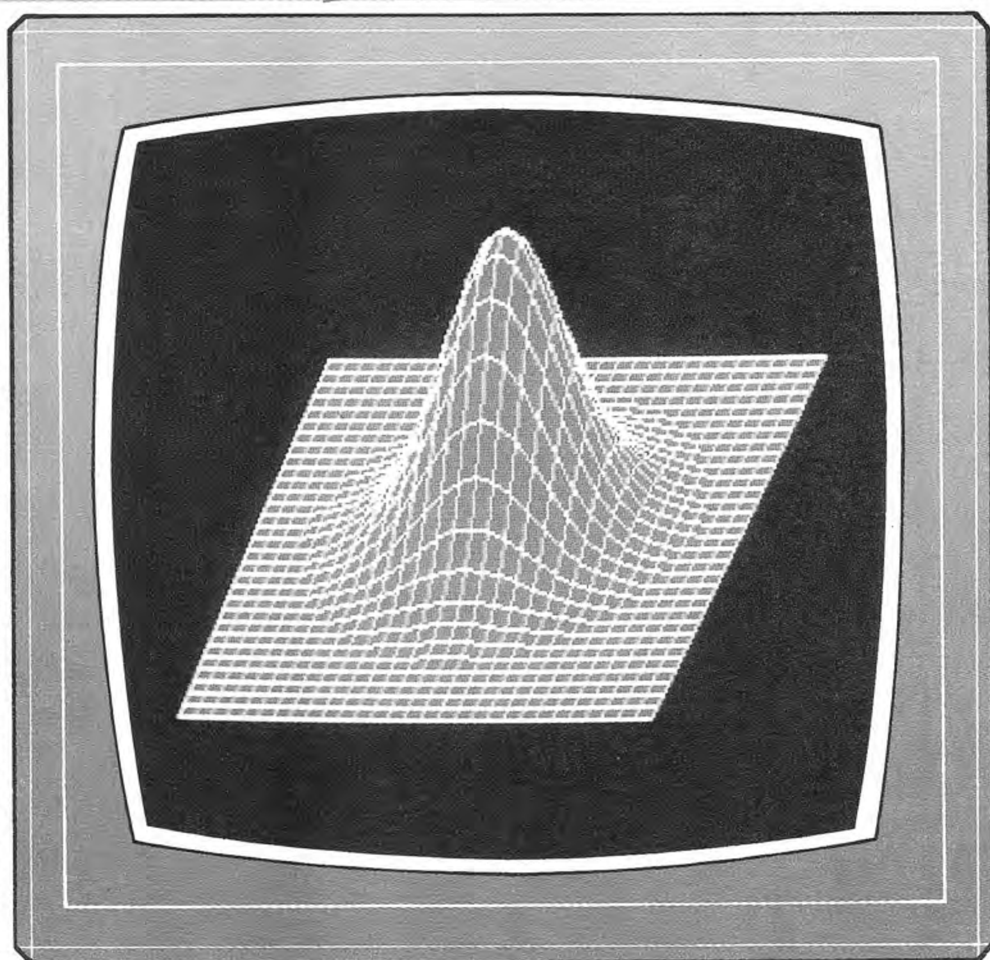
PROGRAMAS EDUCATIVOS
PROGRAMAS DE UTILIDAD
PROGRAMAS DE GESTION
PROGRAMAS DE JUEGOS

▼ BASIC ▼ MAQUINA ▼ PASCAL ▼ LOGO ▼ OTROS LENGUAJES ▼
▼ TECNICAS DE ANALISIS Y DE PROGRAMACION ▼

▼ EDICIONES ▼ SIGLO ▼ CULTURAL ▼

Informática y programación 1

PASO A PASO



PROGRAMAS EDUCATIVOS
PROGRAMAS DE UTILIDAD
PROGRAMAS DE GESTION
PROGRAMAS DE JUEGOS

▼ BASIC ▼ MAQUINA ▼ PASCAL ▼ LOGO ▼ OTROS LENGUAJES ▼
▼ TECNICAS DE ANALISIS Y DE PROGRAMACION ▼

▼ EDICIONES ▼ SIGLO ▼ CULTURAL ▼

Una publicación de

EDICIONES SIGLO CULTURAL, S.A.

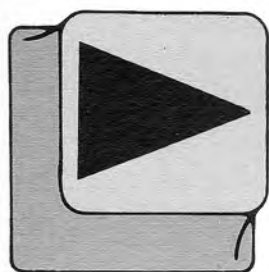
Director-editor:
RICARDO ESPAÑOL CRESPO.
Gerente:
ANTONIO G. CUERPO.
Directora de producción:
MARIA LUISA SUAREZ PEREZ.
Directores de la colección:
MANUEL ALFONSECA, Doctor Ingeniero de Telecomunicación y Licenciado en Informática.
JOSE ARTECHE, Ingeniero de Telecomunicación.
Diseño y maquetación:
BRAVO-LOFISH.
Fotografía:
EQUIPO GALATA.
Dibujos:
JOSE OCHOA

TECNICAS DE PROGRAMACION: Manuel Alfonseca, Doctor Ingeniero de Telecomunicación y Licenciado en Informática. TECNICAS DE ANALISIS: José Arteche, Ingeniero en Telecomunicación. LENGUAJE MAQUINA 8086: Juan Rojas, Licenciado en Ciencias Físicas e Ingeniero Industrial. PASCAL: Juan Ignacio Puyol, Ingeniero Industrial. PROGRAMAS (educativos, de utilidad, de gestión y de juegos): Francisco Morales y colaboradores. Coordinador de Aula de Informática Aplicada (AIA): Alejandro Marcos, Licenciado en Ciencias Químicas. BASIC: Esther Maldonado, Diplomada en Arquitectura. INFORMATICA BASICA: Virginia Muñoz, Diplomada en Informática. LENGUAJE MAQUINA Z-80: Joaquín Salvachúa, Diplomado en Telecomunicación y José Luis Todo, Diplomado en Telecomunicación. LENGUAJE MAQUINA 6502: Jesús Bocho, Licenciado en Informática. LOGO: Cristina Manzanero, Licenciada en Informática. APLICACIONES: Fernando Suero, Diplomado en Telecomunicación. OTROS LENGUAJES (Sistemas operativos): Domingo Villaseñor, Diplomado en Informática, y Lenguaje C: Enrique Serrano, Ingeniero en Telecomunicación.

Ediciones Siglo Cultural, S.A.
Dirección, redacción y administración:
Pedro Teixeira, 8-2.ª planta. Teléf.: 810 52 13. 28020 Madrid.
Publicidad:
Gofar Publicidad, S.A. Benito de Castro, 12 bis. 28028 Madrid.
Distribución en España:
COEDIS, S.A. Valencia, 245. Teléf. 215 70 97. 08007 Barcelona.
Delegación en Madrid: Serrano, 165. Teléf. 411 11 48.
Distribución en Ecuador: Muñoz Hnos.
Distribución en Perú: DISELPESA.
Distribución en Chile: Alfa Ltda.
Importador exclusivo Cono Sur:
CADE, S.R.L. Pasaje Sud América, 1532. Teléf.: 21 24 64.
Buenos Aires - 1.290. Argentina.

ISBN del tomo: 84-7688-069-3
ISBN de la obra: 84-7688-068-7
Fotocomposición:
ARTECOMP, S.A. Albarracín, 50. 28037 Madrid.
Imprime:
MATEU CROMO. Pinto (Madrid).

Depósito legal: M. 5.677-1987
Printed in Spain - Impreso en España.
Suscripciones y números atrasados:
Ediciones Siglo Cultural, S.A.
Pedro Teixeira, 8-2.ª planta. Teléf.: 810 52 13. 28020 Madrid.
Abril, 1987.
P.V.P. Canarias: 335,–.



INDICE

BASIC

INTRODUCCION

EL lenguaje BASIC, cuyo nombre está formado por las iniciales "Begginer's All-purpose Symbolic Instruction Code" (Código Simbólico de Instrucciones de Tipo General para Principiantes), fue desarrollado en la Universidad de Dartmouth (Estados Unidos) por Thomas E. Kurtz y John Kemeny, en la década de los sesenta.

Hasta su aparición, la programación de ordenadores estaba muy restringida a los especialistas, ya que, aparte de la dificultad que entrañaba la programación en sí misma, pocas personas podían tener acceso a máquinas tan costosas.

Kurtz y Kemeny concibieron el lenguaje BASIC con la finalidad de iniciar a los estudiantes en la programación. Su éxito fue enorme dada la facilidad de comunicación con la máquina y la rapidez con que se obtenían los resultados; ello explica, por otra parte, la enorme difusión del lenguaje que ha llegado a ser uno de los más populares al ser implementado en gran número de ordenadores, especialmente en los de tipo doméstico.

El gran avance que supuso el BASIC radica en su naturaleza de lenguaje interpretado, no compilado, lo que supone un gran ahorro de tiempo para el programador.

El BASIC nació para ser un lenguaje fácil de aprender y sencillo de usar. Pero tiene una contrapartida. Al funcionar cada vez en un mayor número de ordenadores de distintas marcas, han aparecido también un gran número de "dialectos", es decir, de distintas versiones del BASIC. Sin embargo, la diferencia entre unas versiones y otras no es muy notable, y en lo fundamental suelen ser todas iguales.

El estudio que se inicia a continuación va dirigido a cubrir las versiones BASIC de un conjunto de ordenadores elegidos entre los de mayor uso en la actualidad.



¿Qué es el Basic?

Todo este conjunto de palabras "raras" constituyen lo que se denomina un programa en BASIC.

Aunque parezca sorprendente este programa nos permite traducir cualquier día de la semana a tres idiomas distintos: inglés, francés y alemán. Primero el ordenador nos preguntará qué día queremos traducir (en español). Una vez que le hayamos dado la respuesta nos da a elegir el idioma que deseamos. Después de hecha la elección el ordenador imprime en pantalla el día que le habíamos dicho, la traducción al idioma elegido y el nombre de dicho idioma. Sorprendente, ¿verdad?

Sin embargo, no se trata de que copiemos en nuestro ordenador este programa sin comprender su funcionamiento. Aun-

que a simple vista pueda parecer que el BASIC es tremendamente complicado, en realidad, es mucho más sencillo de lo que aparenta.

A lo largo de este capítulo y sucesivos vamos a estudiar minuciosamente el lenguaje BASIC de modo que ya no tendremos que limitarnos a copiar programas hechos por otros, sino que podremos crear nuestros propios programas.

Comencemos por ver qué es un programa. Mucha gente cree que los ordenadores son máquinas inteligentes que resuelven todos los problemas. Sin embargo, cuando encendemos un ordenador, la máquina no hace absolutamente nada.

El ordenador no hace nada si no se lo pedimos correctamente. La forma de pedirle algo, de decirle lo que queremos que haga, se llama programa. Por tanto, un programa es un conjunto de instrucciones que introducimos en el ordenador de una forma detallada y ordenada para que realice una determinada tarea. Lo sorprendente es la velocidad con que ejecuta los programas y la capacidad de cálculo que tiene, pero no es, ni mucho menos, una máquina inteligente.

De modo que vamos a pasar ya a ver las diferentes instrucciones BASIC que podemos darle a un ordenador.



PRINT y algo más

La instrucción PRINT sirve para indicarle al ordenador que debe imprimir algo en pantalla (PRINT=imprimir). Pero ¿qué es lo

```
10 REM ***************************
20 REM * PEQUENO DICCIONARIO IBM *
30 REM ***************************
40 CLS
50 DIM E$(7),I$(7),F$(7),A$(7)
60 FOR I=0 TO 7
70 READ E$(I),I$(I),F$(I),A$(I)
80 NEXT I
90 INPUT "¿ QUE DIA DE LA SEMANA QUIERES SABER ? ";R$
100 CLS
110 FOR I=1 TO 7
120 IF R$=E$(I) THEN GOTO 160
130 NEXT I
140 PRINT "ESO NO ES UN DIA DE LA SEMANA"
150 GOTO 90
160 LOCATE 6,17:PRINT "IDIOMAS"
170 LOCATE 7,17:PRINT "-------"
180 LOCATE 10,15:PRINT "1.- INGLES"
190 LOCATE 12,15:PRINT "2.- FRANCES"
200 LOCATE 14,15:PRINT "3.- ALEMAN"
210 LOCATE 20,8:PRINT "PULSA LA OPCION DESEADA"
220 A$=INKEY$:IF A$="" THEN GOTO 220
230 IF ASC(A$)<49 OR ASC(A$)>51 THEN GOTO 220
240 CLS
250 ON VAL(A$) GOSUB 400,500,600
260 LOCATE 20,1:PRINT "¿ QUIERES TRADUCIR OTRO DIA ? (S/N)"
270 A$=INKEY$:IF A$="" THEN GOTO 270
280 IF A$="S" THEN CLS:GOTO 90
290 IF A$<>"N" THEN GOTO 270
300 CLS:END
400 LOCATE 12,1:PRINT R$;" SE ESCRIBE ";I$(I);" EN ";I$(0)
410 RETURN
500 LOCATE 12,1:PRINT R$;" SE ESCRIBE ";F$(I);" EN ";F$(0)
510 RETURN
600 LOCATE 12,1:PRINT R$;" SE ESCRIBE ";A$(I);" EN ";A$(0)
610 RETURN
700 DATA ESPANOL,INGLES,FRANCES,ALEMAN
710 DATA LUNES,MONDAY,LUNDI,MONTAG
720 DATA MARTES,TUESDAY,MARDI,DIENSTAG
730 DATA MIERCOLES,WEDNESDAY,MERCREDI,MITTWOCH
740 DATA JUEVES,THURSDAY,JEUDI,DONNERSTAG
750 DATA VIERNES,FRIDAY,VENDREDI,FREITAG
760 DATA SABADO,SATURDAY,SAMEDI,SAMSTAG
770 DATA DOMINGO,SUNDAY,DIMANCHE,SONNTAG
```

que puede imprimir? Podríamos decir que lo que queramos, pero más concretamente puede imprimir datos numéricos o alfanuméricos. Estos datos pueden ser constantes o variables, sin embargo, de momento vamos a centrarnos en los constantes.

Las constantes numéricas son los números, reales por supuesto, positivos, negativos, enteros o decimales. De todos modos, los números que maneja el ordenador tienen unos límites. La tabla de la figura 1 muestra los números positivos mayor y menor que pueden manejar los principales ordenadores. Los valores están representados en notación científica.

LIMITES DE LOS NUMEROS REALES POSITIVOS		
	Número real positivo máximo	**Número real positivo mínimo**
AMSTRAD	**1.7E + 38**	**1E-38**
COMMODORE	**1.7E + 38**	**2.93873588E-39**
IBM	**10E + 38**	**1E-38**
MSX	**1.7E + 38**	**1E-38**
SPECTRUM	**1E + 38**	**4E-39**

Análogamente, los números negativos también tienen unos límites. En cualquier caso, es muy difícil que alcancemos estos límites en programas normales.

Probemos a teclear la siguiente orden:

```
PRINT 12345
```

Le hemos dicho al ordenador que imprima en pantalla el número 12345; sin embargo, la máquina no hace nada. Para que el ordenador cumpla cualquier orden BASIC debemos pulsar la tecla *INTRO* y el ordenador obedecerá al instante. Hay ordenadores que denominan de forma diferente esta tecla, tal y como podemos ver en la figura, pero su misión es la misma.

DISTINTOS NOMBRES DE LA TECLA INTRO	
AMSTRAD	**INTRO o ENTER**
COMMODORE	**RETURN**
IBM	←┘
MSX	**ENTER o RETURN** o←┘
SPECTRUM	**ENTER**

Por tanto, después de pulsar INTRO ha aparecido en pantalla el número 12345. Sin embargo, esta orden no constituye todavía un programa, sino que es lo que se denomina un comando directo. Los comandos directos se ejecutan en cuanto pulsamos INTRO y no se almacenan en memoria.

Los programas son la forma usual de funcionamiento en BASIC. Cuando trabajamos de este modo, el ordenador va almacenando las órdenes en su memoria cada vez que pulsamos INTRO, y no las ejecutará hasta que no se lo indiquemos de forma expresa.

Para diferenciar una instrucción de un programa de un comando directo basta con poner un número delante de la orden.

Si tecleamos ahora:

```
10 PRINT 12345
```

al pulsar INTRO la orden no se realiza, sino que se almacena en la memoria. Ahora podemos escribir:

```
20 PRINT 67890
```

al pulsar INTRO esta instrucción se almacena en memoria junto a la anterior. De este modo ya tenemos un programa formado por dos instrucciones.

Vamos a realizar un nuevo programa, pero antes debemos borrar el antiguo de la memoria, ya que no podemos almacenar en memoria más de un programa simultáneamente.

Para borrar la memoria tenemos que teclear el comando NEW y a continuación pulsar INTRO. Además, es posible que la pantalla esté llena de cosas. Si queremos borrarla y dejarla limpia no tenemos más que teclear el comando CLS. Algunos ordenadores disponen de la tecla CLEAR para borrar la pantalla.

No debemos confundir NEW con CLS, ya que CLS borra la pantalla pero no la memoria, mientras que NEW borra la memoria, pero no la pantalla (excepto en el SPECTRUM, que también borra la pantalla). Por tanto, NEW es mucho más peligroso.

Comencemos ahora un nuevo programa:

```
10 PRINT "IMPRESION DE CADENAS"
20 PRINT "2468$"
```

Al ejecutar este programa (con el comando RUN) aparecerá en pantalla la frase IMPRESION DE CADENAS y debajo 2468$. Las comillas ya no aparecen, ya que, en realidad, su misión es indicarle al ordenador que está manejando una cadena, y, por tanto, no las imprime.

Finalmente, probemos a borrar la pantalla (CLS). Si ahora queremos volver a ejecutar el programa no tenemos más que teclear otra vez RUN. Sin embargo, si lo que queremos es volver a ver el programa que hemos hecho, lo que se denomina el listado del programa, teclearemos el comando LIST (y luego INTRO). De este modo volverán a aparecer en pantalla todas las instrucciones que constituyen el programa y que están almacenadas en memoria.

RESUMEN
PRINT **RUN** **NEW** **CLS** **LIST**

MAQUINA Z-80

ARQUITECTURA DEL Z-80

(Para los ordenadores Spectrum, Amstrad y MSX)

EL Z-80 es uno de los microprocesadores de 8 bits más usados del mercado. Debido a su versátil conjunto de 158 instrucciones pueden realizarse complejos programas con muy pocas instrucciones.

Su bus de direcciones es de 16 bits, por lo que será capaz de direccionar 65.536 posiciones distintas de memoria (64 Kbytes). Como cada posición de memoria es de sólo 8 bits, para almacenar una dirección en memoria se necesitarán dos palabras consecutivas de 8 bits.

Como, en general, cada instrucción opera sobre alguna posición de memoria, deberá llevar la dirección de dicha posición. Por tanto, las instrucciones se compondrán de dos campos distintos: el campo de operación, que indica la instrucción a ejecutar, y el campo de dirección, que indica la posición de memoria con que se desea operar. Según la longitud de estos distintos campos, así será la longitud de la instrucción, por lo que existirán instrucciones que ocuparon 1, 2, 3 ó 4 bytes.

Además de la memoria externa existen en la CPU una serie de registros.

Estos registros están divididos en dos partes: registros de uso general y registros especiales.

Los registros de uso general son los utilizables por el usuario.

De éstos existen dos juegos, pero para operar sólo se utiliza el juego principal. El uso del juego alternativo es para realizar copias de todos los registros mediante una única instrucción.

Estos registros pueden usarse como 8 de 8 bits (A,F,B,C,D,E,H,L,) o como 4 de 16 bits (AF, BC, DE, HL).

Los registros especiales son los usados por la CPU para almacenar sus datos.

El registro A es el usado como acumulador por la CPU.

El registro F es el de estado de la CPU que contiene las siguientes banderas, o flags:

S: Signo.
Z: Cero.
X: Sin significado especial.
H: Semiarrastre.
P/V: Paridad/Desbordamiento.
C: Arrastre.

Los registros especiales son los usados por la CPU para sus datos. Estos son:

— El contador de programa (PC) de 16 bits.
— El puntero de la pila (SP) de 16 bits.
— Dos registros índice para direccionamiento indexado de 16 bits cada uno.
— Un registro de interrupción de 8 bits.
— Un registro de refresco de memoria usado por el hardware de la CPU.

Juego principal		Juego alternativo		
Acumulador A	Flags. F	Acumulador A'	Flags. F'	Registros generales
B	C	B'	C'	
D	E	D'	E'	
H	L	H'	L'	

Vector interrupciones I	Refresco memoria R	
Registro índice IX		Registros especiales
Registro índice IY		
Puntero del stack SP		
Contador de programa PC		

El significado, uso y misión de las banderas y los registros indicados anteriormente se verá con más detalle al explicar, en otros capítulos, las diversas instrucciones.

 Conjunto de instrucciones del microprocesador Z-80

A continuación proporcionamos al lector una tabla con todas las microinstrucciones del Z-80 y una explicación simbólica de cada una de ellas, que será de gran utilidad a la hora de programar en ensamblador. Por eso aconsejamos que se tenga presente esta tabla en futuros apartados de lenguaje máquina de Z-80.

Las instrucciones están clasificadas en los grupos siguientes:

— Cargas de 8 bits.
— Cargas de 16 bits.
— Intercambios.
— Movimiento de bloques de memoria.
— Búsqueda de bloques de memoria.
— Aritmética y lógica de 8 bits.
— Aritmética y lógica de 16 bits.
— Operaciones de uso general en acumulador con banderas.
— Instrucciones varias.
— Rotaciones y desplazamientos.
— Comprobación y puestas a cero y uno de bits.
— Entradas y salidas.
— Saltos.
— Llamadas.
— Inicializaciones.
— Retornos.

En la tabla se usan los siguientes símbolos:

*b = un n.° binario en un registro de 8 bits o de una posición de memoria.

*CC = Códigos de estado de las banderas (Flags).
NZ = No es cero.
Z = Es cero.
NC = No hay "carry".
C = Hay "carry".
PO = Paridad impar / no desbordamiento.
PE = Paridad par / desbordamiento (overflow).
P = Valor positivo.
M = Negativo (M → menos).
*d = Destino de 8 bits (registro o posición de memoria).
*dd = Destino de 16 bits (registro o memoria).
*e = Desplazamiento de 8 bits en complemento a 2 con signo utilizado en saltos relativos y direccionamiento indexado.
*L = 8 posiciones especiales de memoria de la página cero.

En notación decimal son 0, 8, 16, 24, 32, 40, 48 y 56.

*n = Cualquier número de 8 bits.

*nn = Cualquier número de 16 bits.
*r = Cualquier registro de propósito general de 8 bits (A,B,C,D,E,H, ó L).
*S = Cualquier registro de 8 bits o posición de memoria como fuente.
*Sb = Un bit en un registro de 8 bits o posición de memoria específicos.
*SS = Cualquier registro de 16 bits o 2 posiciones de memoria como fuente.
*Suscrito con "L"= Los 8 bits menos significativos de un registro de 16 bits.
*Suscrito con "H" = Los 8 bits más significativos de un registro de 16 bits.
*( )= El contenido entre paréntesis es utilizado como puntero de una dirección de memoria o de E/S.
**Los registros de 8 bits son:

A,B,C,D,E,H,L,I y R.

**Los pares de 2 registros (16 bits) son:

AF,BC,DE y HL

**Los registros de 16 bits son:

SP,PC,IX e IY

**Los modos de direccionamiento utilizados incluyen combinaciones de los siguientes:

— Inmediato.
— Inmediato extendido.
— Modificando página cero.
— Relativo.
— Extendido.
— Indexado.
— Registro.
— Implicado.
— Indirecto con registro.
— Bit.

Juego principal		Juego alternativo		
Acumulador *A*	Flags *F*	Acumulador *A'*	Flags *F'*	
B	*C*	*B'*	*C'*	Registros generales
D	*E*	*D'*	*E'*	
H	*L*	*H'*	*L'*	

Vector interrupciones *I*	Refresco memoria *R*	Registros especiales
Registro índice *IX*		
Registro índice *IY*		
Puntero del stack *SP*		
Contador de programa *PC*		

	Mnemotécnico	Operación	Comentario
CARGAS DE 8 BITS	LD r,s	r←s	s=r,n (HL) (IX+e) , (IY+e)
	LD d,r	d←r	d=(HL),r (IX+e),(IY+e)
	LD d,n	d←n	d=(HL), (IX+e),(IY+e)
	LD A,s	A←s	s=(BC),(DE), (nn),I,R,
	LD d,A	d←A	d=(BC),(DE). (nn),I,R,
CARGAS DE 16 BITS	LD dd,nn	dd←nn	dd=BC,DE, HL,SP,IX,IY
	LD dd,(nn)	dd←(nn)	dd=BC,DE, HL,SP,IX,IY
	LD(nn),ss	(nn)←ss	ss=BC,DE, HL,SP,IX,IY,
	LD SP,ss	SP←ss	ss=BC,DE,
	PUSH ss	(SP-1)←ss_H (SP-2)←ss	HL,AF,IX,IY
	POP dd	dd_L←(SP),dd_H←(SP+1)	dd=BC,DE, HL,AF,IX,IY
INTERCAMBIOS	EX DE,HL	DE←HL	
	EX,AF,AF	AF←AF	
	EXX	(BC DE HL)←(BC' DE' HL')	
	EX (SP),ss	(SP)←ss_L (SP+1)←ss_H	ss=HL,IX,IY
MOV DE BLOQUES	LDI	(DE)←(HL),DE←DE+1 HL←HL+1,BC←BC-1	
	LDIR	(DE)—(HL),DE←DE+1 HL←HL+1,BC←BC-1 Repite hasta BC=0	
	LDD	(DE)←(HL),DE←DE-1 HL←HL-1,BC←BC-1 Repite hasta BC=0	
BUSQUEDA DE BLOQUES	CPI	A-(HL),HL←HL+1 BC←BC-1	
	CPIR	A-(HL),HL←HL+1 BC←BC-1, Repite hasta BC=0 o A=(HL)	A-(HL) Sólo afecta a los flags, no a A
	CPD	A-(HL),HL←HL-1 BC←BC-1	
	CPDR	A-(HL),HL←HL-1 BC←BC-1, Repite hasta BC=0 o A=(HL)	
OPERACIONES DE 8 BITS	ADD s	A←A+s	CY es el flag del carry s=r,n,(HL) (IX+e),(IY+e)
	ADC s	A←A+s+CY	
	SUB s	A←A-s	
	SBC s	A←A-s-CY	
	AND s	A←A s	
	OR s	A←A s	
	XOR s	A←A s	
	CP s	A←s	s=r,n,(HL) (IX+e), (IY+e)
	INC d	d←d+1	d=r,(HL) (IX+e),(IY+e)
	DLC d	d←d-1	
BIT S,R y T	BIT b,s	Z←s_b	Z es el flag de cero s=t,(HL) (IX+e),(IY+e)
	SET b,s	s_b←1	
	RES b,s	s_b←0	

	Mnemotécnico	Operación	Comentario
OPERACIONES 16 BITS	ADD HL,ss	HL←HL+ss	ss=BC,DE HL,SP
	ADC HL,ss	HL←HL+ss+CY	
	SBC HL,ss	HL←HL-ss-CY	
	ADD IX,ss	IX←IX+ss	ss=BC.DE IX,SP
	ADD IY,ss	IY←IY+ss	ss=BC,DE IY,SP
	INC dd	dd←dd+1	dd=BC,DE HL,SP,IX,IY
	DEC,dd	dd←dd-1	dd=BC,DE, HL,SP,IX,IY
OPS. A y FLAGS	DAA	Convierte el cont. de A en formato BCD y luego suma o resta.	El operando debe estar en formato BCD.
	CPL	A←Ā	
	NLG	A←00-A	
	CCF	CY←$\overline{CY}$	
	SCF	CY←I	
INS. VARIAS	NOP	No operar	
	HALT	Para la CPU	
	DI	Desactiva interrup.	
	II	Activa interrupciones	
	IM 0	Activa modo INT.0	8080A MODO
	IM 1	Activa modo INT.1	Llama 0038
	IM 2	Activa modo INT.2	Llamada directamente
ROTACIONES Y DESPLAZAMIENTOS	RLC s		s=R,(HL) (IX+e),(IY+e)
	RL s		
	RRC s		
	RR s		
	SLA s		
	SRA s		
	SRL s		
	RLD		
	RRD		
ENTRADA/SALIDA	IN A,(n)	A←(n)	
	IN c,(C)	r—(C)	Activa flags
	INI	(HL)←(C),HL←HL+1 B←B-1	
	INIR	(HL)←(C),HL←HL+1 B←B-1 Repetir hasta B=0	
	IND	(HL)←(C),HL←HL-1 B←B-1	
	INDR	(HL)←(C),HL←HL-1 B←B-1 Repetir hasta B=0	
	OUT(n),A	(n)←A	
	OUT(C),r	(C)←r	
	OUTI	(C)←(HL),HL←HL+1 B←B-1	
	OTIR	(C)←(HL),HL←HL+1 B←B-1	
	OTDR	(C)←(HL),HL←HL-1 B←B-1 Repetir hasta B=0	
SALTOS	JP nn	PC←nn	cc {NZ PO, Z PE, NC P, C M}
	JP cc,nn	Si condición es verdad PC←nn,sino continuar	
	JR e	PC←PC+e	
	JR kk,e	Si condición kk es verdad PC←PC+e, sino continuar	kk {NZ NC, Z C}
	JP (ss)	PC←ss	ss=HL,IX,IY
	DINZ e	B←B-1 Si B=0 continuar, sino PC←PC+e	
LLAMADAS	CALL nn	(SP-1)←PC_H (SP-2)←PC_L,PC←nn	cc {NZ PO, Z PE, NC P, C M}
	CALL cc,nn	Si condición cc es falsa continuar, sino c igual que CALL nn	
INICIA.	RST L	(SP-1)←PC_H (SP-2)←PC_L,PC_H←0 PC_L←L	
RETORNO	RET	PC_L←(SP) PC_H←(SP+1)	cc {NZ PO, Z PL, NC P, C M}
	RET cc	Si condi. cc es falsa continuar, sino a igual a RET	
	RETI	Retorno de inter.=RET	
	RETN	Retorno de interrupción no mascarable	

MAQUINA 8086



Todo lo tratado en esta sección es válido para la familia de microprocesadores usados en los IBM PC que son los 8088, 8086 y 80286 de Intel. En adelante, y por simplicidad, nos referiremos al 8088 como representativo de toda la familia.

LOS lenguajes de alto nivel como BASIC, PASCAL, FORTRAN, etc., se caracterizan por tener instrucciones muy potentes con las cuales se pueden programar tareas complicadas de forma relativamente sencilla. Otra característica de estos lenguajes es que son (o tienden a ser) independientes de las máquinas.

Estas dos características, muy deseables para los usuarios, tienen como contrapartida el hecho de que la velocidad de las tareas programadas en estos lenguajes es considerablemente menor que la velocidad que obtendríamos programando las mismas tareas en Lenguaje Máquina. Esto es debido a que los compiladores e intérpretes están diseñados para los casos generales y no son capaces de emplear el conjunto de instrucciones máquina más idóneo para ejecutar cada tarea concreta.

Por otra parte, la mayoría de los lenguajes de alto nivel no contemplan siquiera la posibilidad de ejecutar algunas tareas, como pueden ser, una determinada gestión de la memoria o de los dispositivos externos.

Los usuarios que se encuentren con estas dificultades, o los que simplemente quieran conocer con detalle cómo es el funcionamiento interno del ordenador, son usuarios potenciales del Lenguaje Máquina.



¿Qué es el Lenguaje Máquina?

Estrictamente hablando, el Lenguaje Máquina es el lenguaje que entiende directamente el microprocesador. La CPU (unidad central de proceso) del IBM PC, que es el microprocesador 8088 de Intel, reconoce un centenar de instrucciones diferentes. Cada instrucción es un grupo de 1 a 7 posiciones de memoria (bytes).

Veamos un ejemplo: los siguientes 20 bytes constituyen un programa en Lenguaje Máquina que escribe repetidamente el carácter «A» sobre la pantalla:

1E 31 C0 50 B8 00 B0 8E C0 31 FF B9 C0 03
B8 41 F1 F3 AB CB

Ningún programa escrito en un lenguaje de alto nivel puede realizar esta tarea de forma más rápida y concisa. Pero ¿cuánto tiempo puede tardarse en escribir programas en este lenguaje? y ¿cuántos errores pueden cometerse transcribiendo programas de este tipo?

Los pioneros de la informática programaban de esta forma, pero afortunadamente pronto se les ocurrió la idea de emplear códigos nemotécnicos que sustituyeran a los códigos que entiende la máquina y que fueran más fáciles de recordar. A esto se le conoce como «Lenguaje Simbólico». El programa que reconoce los códigos nemotécnicos del lenguaje simbólico y los traduce al Lengua-

je Máquina recibe el nombre de «Programa Ensamblador».

Debido a la estrecha relación que existe entre estos conceptos, se suelen emplear como sinónimos los términos de «Lenguaje Máquina», «Lenguaje Simbólico» y «Lenguaje Ensamblador».



Código Máquina

Por una vez, podemos emular a los pioneros y escribir un programa directamente en código máquina. Para ello contamos con el programa DEBUG suministrado en el diskette del sistema operativo DOS. Si consultamos un diccionario de lengua inglesa, nos encontramos con que el término «debug» significa «buscar y eliminar posibles causas de problemas, fallos y errores». Pues bien, este programa sirve exactamente para eso, y es una ayuda prácticamente indispensable para hacer que funcionen correctamente los programas escritos en Lenguaje Máquina. Su uso nos va a facilitar la comprensión de cómo el microprocesador ejecuta las instrucciones que le hemos programado.

Cuando desde DOS pedimos la ejecución del DEBUG, aparece en la línea siguiente un guión, que es el símbolo con el que este programa nos indica que está esperando órdenes del usuario.

```
A>DEBUG
```

El programa es conversacional y admite 17 órdenes o «mandatos» diferentes, los cuales se especifican con una letra (la inicial de la expresión inglesa del mandato), seguida (opcionalmente) de un espacio en blanco y de unos parámetros. Los parámetros, que pueden ser muy variados, se expresan siempre que son numéricos en notación hexadecimal.

Para los que no estén familiarizados con esta forma de expresar los números, diremos que al igual que en la notación binaria se necesitan dos dígitos (el 0 y el 1) y en la notación decimal se necesitan 10 (0, 1, 2, 3, 4, 5, 6, 7, 8 y 9), en la notación hexadecimal se necesitan 16 dígitos diferentes. El problema de los 6 dígitos que faltaban se ha resuelto escogiendo las 6 primeras letras del abecedario, (en vez de inventar símbolos nuevos). De esta forma las letras A, B, C, D, E, y F representan en esta notación los números decimales 10, 11, 12, 13, 14 y 15 respectivamente. En hexadecimal los números del 1 al 50 se escriben así:

1 2 3 4 5 6 7 8 9 A B C D E F 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F 20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F 30 31 32

El DEBUG considera equivalentes a todos los efectos las letras mayúsculas y minúsculas.

La escritura del programa en Lenguaje Máquina la realizaremos en 6 pasos, en los que utilizaremos los siguientes mandatos del DEBUG:

- E (inicial de ENTER) para introducir datos en memoria.
- D (inicial de DISPLAY) para visualizar parte de la memoria.
- N (inicial de NAME) para definir el nombre de un fichero.
- R (inicial de REGISTER) para visualizar y modificar el valor de los registros.
- W (inicial de WRITE) para escribir un fichero en diskette.
- Q (inicial de QUIT) para terminar la ejecución y volver al DOS.

Paso primero: escritura de 20 números hexadecimales (cada uno de los cuales ocupará un byte) a partir de la dirección 100 (hexadecimal) de la memoria. Escribimos "e 100 ", a continuación los 20 números hexadecimales que constituyen nuestro programa y como siempre terminamos pulsando la tecla Enter.

Antes de hacerlo, tengamos en cuenta que el byte número 7 del programa está relacionado con el tipo de pantalla a usar. Si se utiliza la pantalla monocroma, dicho byte debe ser «b0», y si se utiliza la de color debe ser «b8».

```
-e 100 1e 31 c0 50 b8 00 xx 8e c0 31 ff
   b9 c0 03 b8 41 f1 f3 ab cb
```

donde xx = b0 (pantalla monócroma)
b8 (pantalla de color)

Paso segundo: comprobación de que los datos definidos son exactamente los que queríamos. Para ello usamos el mandato "d 100 113" que quiere decir visualizar en la pantalla desde la dirección 100 a la 113. Al hacer esto obtendremos en pantalla lo siguiente:

```
-d 100 113
0C47:0100 1E 31 C0 50 B8 00 B0 8E-C0 31
          FF B9 C0 03 B8 41. 1 P8.0. 1.9.8A
0C47:0110 F1 F3 AB CB                ps+K
```

La salida producida por este mandato está dividida en tres partes. A la izquierda aparecen las direcciones de memoria que se están representando (el código 0C47 que aparece a la izquierda, puede variar y no debe preocuparnos). En el centro se representa el contenido de los bytes pedidos en notación hexadecimal y a la derecha la representación ASCII de los mismos bytes.

Debemos comprobar que los números que aparecen en la parte central son byte por byte exactamente los que queríamos. Si coinciden, podemos continuar con el tercer paso, pero si se encuentra algún error hay que volver a repetir el primer paso, ya que la ejecución de programas en Lenguaje Máquina con errores obliga en la mayoría de los casos a hacer «IPL» (arrancar de nuevo).

Paso tercero: definición del nombre del programa. Para ello escribimos el mandato «N», después un espacio y a continuación el nombre elegido seguido de un punto y la palabra «COM». Si elegimos por ejemplo, el nombre «MAQUINA» escribiremos "n maquina.com", presionaremos ENTER y en la pantalla deberá aparecer:

```
-n maquina.com
```

Paso cuarto: definición de la longitud del programa que queremos guardar. El DEBUG necesita tenerla en el registro CX. Para definir un 20 (decimal) en ese registro, tecleamos "r cx" seguido de Enter. El DEBUG responde con CX seguido del valor que en ese momento contiene el registro y en la línea siguiente escribe dos puntos y espera que se especifique un nuevo valor para CX (en hexadecimal). Nosotros escribiremos «14» (puesto que 14 hexadecimal es lo mismo que 20 decimal) y pulsaremos ENTER.

```
-r cx
CX 0000
:14
```

Paso quinto: escritura del fichero en el diskette constituyendo un programa ejecutable. Utilizamos el mandato W sin parámetros. El DEBUG devolverá un mensaje que confirma el número de bytes escritos (en notación hexadecimal).

```
-w
Writing 0014 bytes
```

Paso sexto: terminación del DEBUG y vuelta al DOS. Esto lo hacemos con el mandato Q sin parámetros.

```
-q
A>
```

Si listamos ahora el directorio del diskette, podemos comprobar que tenemos un nuevo fichero de sólo 20 bytes de nombre MAQUINA.COM. Para ejecutarlo, escribimos su nombre (sin el punto ni la palabra COM) seguido de ENTER:

```
A>MAQUINA
```

El programa escribirá parpadeando sobre la pantalla 960 veces el carácter «A». Como curiosidad podemos señalar que dicho carácter está definido en el byte número 16 del programa y el atributo en el byte siguiente.

Este ejemplo nos ha servido para comprobar directamente lo árido que resulta el código máquina y para empezar a familiarizarnos con el DEBUG. Ahora nos va a volver a servir para comprender el concepto del lenguaje simbólico.



Lenguaje Simbólico

Volvemos a llamar de nuevo al DEBUG, esta vez usando como parámetro el nombre del programa que acabamos de crear.

```
A>DEBUG MAQUINA.COM
```

Y ejecutamos un nuevo mandato, el mandato U. Esta letra es la inicial de la palabra inglesa «unassemble» que quiere decir des-ensamblar y que nosotros podemos traducir por «deshacer la codificación que hizo el ensamblador», ya que la forma normal de obtener el código máquina es a través del programa ensamblador aunque nosotros lo hayamos escrito directamente en este caso. Escribimos "u 100 113", con lo cual pedimos al DEBUG que nos represente las instrucciones del «lenguaje simbólico» que contienen las posiciones de memoria comprendidas entre las direcciones 100 y 113 (hexadecimal). Al pulsar Enter obtendremos:

```
-u 100 113

0C47:0100   1E       PUSH   DS
0C47:0101   31C0     XOR    AX,AX
0C47:0103   50       PUSH   AX
0C47:0104   B800B0   MOV    AX,B000
0C47:0107   8EC0     MOV    ES,AX
0C47:0109   31FF     XOR    DI,DI
0C47:010B   B9C003   MOV    CX,03C0
0C47:010E   B841F1   MOV    AX,F141
0C47:0111   F3       REPZ
0C47:0112   AB       STOSW
0C47:0113   CB       RETF
<---------> <------> <------------------>
  zona 1     zona 2        zona 3
```

El resultado del mandato U está dividido en tres zonas. La zona de la izquierda contiene direcciones de memoria, la zona central representa el contenido de esa memoria en notación hexadecimal (podemos reconocer en las distintas líneas la lista de códigos que anteriormente habíamos definido con el mandato E) y en la zona de la derecha se representa el programa en «lenguaje simbólico» asociado a dichos códigos.

Llegamos a la conclusión que ya habíamos anticipado anteriormente. El código simbólico y el código máquina son dos formas diferentes de representar lo mismo. El primero está adaptado al hombre y el segundo está adaptado a la máquina, pero existe entre ambos una correspondencia y hay programas para realizar conversiones en los dos sentidos.

En el listado del programa que nos ha proporcionado el DEBUG, observamos dos columnas. La columna de la izquierda contiene palabras como PUSH, XOR, MOV, etc., que son códigos nemotécnicos de instrucciones. Lo que aparece en la columna de la derecha son operandos de dichas instrucciones. Podemos ver operandos de dos tipos, números hexadecimales como B000, 03C0 y F141 y otras palabras: DS, AX, ES, DI y CX que son códigos nemotécnicos de registros y es de esto de lo que vamos a tratar en el tomo cuatro.

PROGRAMAS

PROGRAMAS EDUCATIVOS
PROGRAMAS DE UTILIDAD
PROGRAMAS DE GESTION
PROGRAMAS DE JUEGOS

EN esta sección iremos viendo semana tras semana una serie de programas que, no sólo nos servirán para jugar, sino también para aprender matemáticas, física, química, para llevar nuestra propia contabilidad casera, nuestro fichero de libros... También veremos programas de ayuda al programador y rutinas, en BASIC y en CODIGO MAQUINA, con las que sacar más provecho a nuestro ordenador:



Programa: Cálculo de determinantes

El primer programa que vamos a ver sirve para resolver determinantes de cualquier orden. La velocidad de este programa es sorprendente aunque tenga que resolver determinantes de 20 x 20 (o de 40 x 40, que es lo máximo que acepta).

```
DDETERMINANTES
==============

100 REM   ************************************************************
101 REM   ********  CALCULO DE DETERMINANTES DE CUALQUIER ORDEN   **********
102 REM   ************************************************************
103 REM
104 REM   ************************************************************
105 REM   *************  POR JUAN MANUEL GUTIERREZ LEITON   **************
106 REM   ************************************************************
107 REM
108 REM   ************************************************************
109 REM   *************  (c) EDICIONES SIGLO CULTURAL,1987  ***************
110 REM   ************************************************************
111 REM
112 DIM M(1600),D(1600)
113 CLS
114 INPUT "dimension del determinante";DI
115 FOR F=1 TO DI
116    FOR C=1 TO DI
117       PRINT "A(";F;",";C;")=";
118       LET D=C-DI+F*DI
119       INPUT M(D)
120    NEXT C
121 NEXT F
122 LET E=0
123 LET F=0
124 FOR I=1 TO (DI-1)*DI-1 STEP DI+1
125    LET E=E+1
126    LET C=I
```

```
127     IF M(C)=0 THEN LET F=1:LET C=C+DI:GOTO 130
128     IF F=1 THEN GOTO 132
129     GOTO 136
130     IF C<DI^2 THEN GOTO 127
131     PRINT "determinante=0":GOTO 161
132     FOR G=I TO I+DI-E
133        LET M(G)=M(G)+M(C)
134        LET C=C+1
135     NEXT G
136     LET D(E)=M(I)
137     FOR G=I TO E*DI
138        LET M(G)=M(G)/D(E)
139     NEXT G
140     LET H=E
141     LET J=I+DI
142     LET K=M(J)
143     LET P=J
144     FOR G=I TO I+DI-E
145        LET M(P)=M(P)-M(G)*K
146        LET P=P+1
147     NEXT G
148     LET J=J+DI
149     IF H<DI-1. THEN LET H=H+1:GOTO 142
150 NEXT I
151 LET D(E+1)=M(DI*DI)
152 LET RE=1
153 FOR I=1 TO DI
154     LET RE=RE*D(I)
155 NEXT I
156 CLS
157 PRINT "el determinante=";RE
158 LOCATE 12,20
159 PRINT "pulse una tecla para continuar"
160 A$=INKEY$:IF A$="" THEN GOTO 160
161 CLS
162 LOCATE 1,1
163 PRINT "CALCULAR OTRO DETERMINANTE (S/N)";
164 LOCATE 1,33:INPUT A$
165 IF A$="S" OR A$="s" THEN GOTO 113
166 IF A$="N" OR A$="n" THEN END
167 LOCATE 12,20:PRINT "caracter no valido":GOTO 162
```

Este programa funciona perfectamente en el IBM PC y compatibles, así como en el AMSTRAD. Para que el programa pueda funcionar en otros ordenadores, proponemos las siguientes modificaciones:

COMMODORE:

```
113 PRINT CHR$(147)
156 PRINT CHR$(147)
158 POKE 214,12:POKE 211,1
161 PRINT CHR$(147)
162 POKE 214,1:POKE 211,1
167 POKE 214,12:POKE 211,1
```

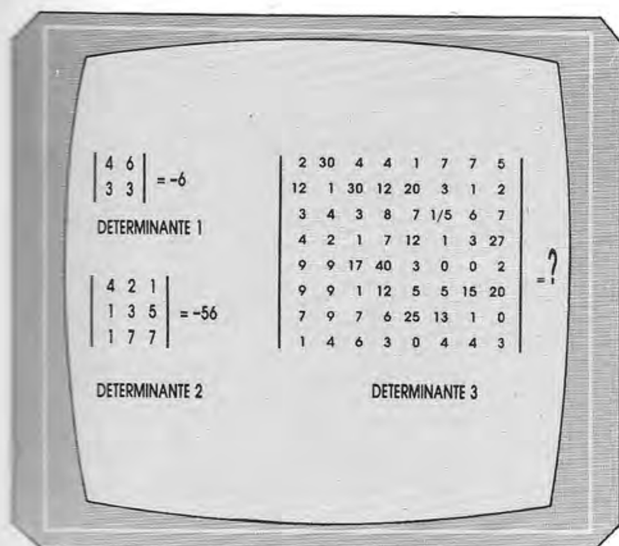
MSX:

```
158 LOCATE 1,12
162 LOCATE 1,1
167 LOCATE 1,12
```

SPECTRUM:

```
119 INPUT M(D):PRINT M(D)
158 PRINT AT 12,1;
160 LET A$=INKEY$:IF A$="" THEN GOTO 160
164 INPUT "LO CALCULAMOS";A$
167 PRINT AT 12,1; "Caracter no valido":GOTO 162
```

Este programa es muy fácil de pasar a una calculadora programable debido a su pequeña longitud. Para ello te aconsejo que leas el manual de tu calculadora aunque no creo que tengas que hacer muchos cambios. Sólo necesitarás anular todos las sentencias LOCATE del listado.





Resolver un determinante como el 3 a mano es una labor casi imposible. Con este programa verás como tu ordenador lo resuelve en menos de 15 segundos.



Notas sobre el programa 1

Este programa está preparado para hacer determinantes de un máximo de 40 × 40. Si quieres hacer determinantes más grandes tienes que cambiar el dimensionado de los vectores de la línea 112. Por ejemplo, si necesitas hacer un determinante de 48 × 48 la línea 112 nos quedaría:

112 DIM M(2304):DIM D(2304)p

El número 2304 nos resulta de multiplicar 48 × 48.

Si piensas unir este programa con algún otro que tú tengas, y quieres ahorrar memoria, porque no necesitas realizar determinantes de 40 × 40, te aconsejo que varíes la línea 112 y ajustes la dimensión de los vectores a lo que necesites.



Programa: Vocabulario inglés

Este programa nos va a permitir aprender algo más de inglés del que ya sabemos. Está pensado para que una persona que tenga pocos conocimientos de dicho idioma pueda aprender poco a poco con él. Pero también sirve para aquéllos que quieran recordar todo lo que aprendieron hace tiempo y que casi han olvidado.

```
VOCABULARIO INGLES
------------------

100 REM **********************************
101 REM *                                *
102 REM *       ENGLISH VOCABULARY       *
103 REM *                                *
104 REM * Por PETER BERGMANN             *
105 REM **********************************
106 REM
107 REM ************************************
108 REM *                                  *
109 REM *         VARIABLES USADAS         *
110 REM *         ----------------         *
111 REM *                                  *
112 REM * S$ = PALABRA ESPA%OL             *
113 REM * I$ = PALABRA INGLES              *
114 REM * C  = RESPUESTAS (CORR. O FAL)    *
115 REM * U  = PALABRA USADA FLAG          *
116 REM * NN = NUMERO DE NIVEL             *
117 REM * NC = NUMERO DE RESP. CORRECTO    *
118 REM * NF = NUMERO DE RESP. FALSO       *
119 REM * PC = % CORRECTO                  *
120 REM * PF = % FALSO                     *
121 REM * RN = NUMERO ALEATORIO            *
122 REM * NL = NUMERO DE LETRAS            *
123 REM * R$ = RESPUESTAS (S/N)            *
124 REM * N1, N2, I = CONTADORES           *
125 REM ************************************
126 REM
127 DIM S$(50),I$(50)
128 DIM C(50),U(50)
129 REM
130 REM ***  PANTALLA DE PRESENTACION ***
131 REM
132 CLS
```

```
133 PRINT "*****************************************"
134 FOR I=2 TO 19
135    PRINT "*";TAB(40);"*"
136 NEXT I
137 PRINT "*****************************************"
138 LOCATE 8,13:PRINT "ENGLISH VOCABULARY"
139 LOCATE 9,12:PRINT "--------------------"
140 LOCATE 12,7:PRINT "(c) Ed. Siglo Cultural, 1987"
141 FOR I = 1 TO 1000
142 NEXT I
143 REM
144 REM *** ENTRADA DE NIVEL ***
145 REM
146 LET NP=0
147 LOCATE 17,3
148 INPUT "(QUE NIVEL QUIERES  (1-4) ";NN
149 IF NN<1 OR NN>4 THEN GOTO 148
150 RANDOMIZE TIMER
151 LET NP=0:LET NC=0:LET NF=0
152    FOR I=1 TO 50
153    LET C(I)=0
154    LET U(I)= 0
155 NEXT I
156 REM
157 REM  *** LECTURA DE LA DATA REQUERIDA ***
158 REM
159 LET NP=NP+1
160 IF NP=51 THEN GOTO 304
161 IF NN=1 THEN RESTORE 323
162 IF NN=2 THEN RESTORE 344
163 IF NN=3 THEN RESTORE 399
164 IF NN=4 THEN RESTORE 454
165 REM
166 REM *** LECTURA DE LAS PALABRAS ***
167 REM
168 LET RN=INT(RND*50+1)
169 FOR I=1 TO RN
170    READ S$(NP),I$(NP)
171 NEXT I
172 IF U(RN)<>0 THEN GOTO 161
173 LET U(RN)=1
174 CLS:PRINT:PRINT
175 PRINT "La palabra numero ";NP;" es : ";S$(NP)
176 PRINT
177 PRINT
178 PRINT "ESCRIBE ESTA PALABRA EN INGLES"
179 PRINT
180 LET NL=LEN(I$(NP))
181 LOCATE 11,INT((40-NL)/2)
182 FOR I=1 TO NL
183    PRINT "-";
184 NEXT I
185 LOCATE 10,INT((40-NL)/2)-2
186 INPUT E$
187 CLS
188 PRINT "MI PREGUNTA FUE ";S$(NP)
189 PRINT
190 PRINT
191 PRINT "TU RESPUESTA ES ";E$
192 PRINT
193 PRINT
194 IF E$=I$(NP) THEN GOTO 197
195 GOTO 219
196 REM
197 REM *** RESPUESTA CORRECTA ***
198 REM
199 IF C(NP)<>0 THEN GOTO 201
200 LET C(NP)=1
201 PRINT
202 FOR I=1 TO 10
203    LOCATE 12,15
204    PRINT "HAS ACERTADO"
205    FOR J=1 TO 100-3*I
206    NEXT J
207    LOCATE 12,15
208    PRINT "            "
209    FOR J=1 TO 100-3*I
210    NEXT J
211 NEXT I
212 LOCATE 15,1
213 PRINT "(QUIERES OTRA PALABRA? (S/N) "
214 INPUT R$
```

```
215 IF R$ = "S" THEN GOTO 157
216 IF R$ = "N" THEN GOTO 242
217 GOTO 212
218 REM
219 REM *** RESPUESTA ERRONEA ***
220 REM
221 IF C(NP)<>0 THEN GOTO 223
222 LET C(NP)=2
223 FOR I=1 TO 7
224     LOCATE 12,8
225     PRINT "LO SIENTO. NO HAS ACERTADO"
226     FOR J=1 TO 40+3*I
227     NEXT J
228     LOCATE 12,8
229     PRINT "                          "
230     FOR J=1 TO 40+3*I
231     NEXT J
232 NEXT I
233 LOCATE 13,1
234 INPUT "(QUIERES INTENTARLO OTRA VEZ (S/N) ";R$
235 IF R$="S" THEN GOTU 174
236 IF R$<>"N" THEN GOTO 233
237 LOCATE 15,1
238 PRINT "LA RESPUESTA ERA: ";I$(NP)
239 FOR I=1 TO 2000
240 NEXT I
241 GOTO 159
242 REM
243 REM ***  CALCULO DE % DE ACIERTOS ***
244 REM
245 FOR I=1 TO NP
246     IF C(I)=1 THEN NC=NC+1
247     IF C(I)=2 THEN NF=NF+1
248 NEXT I
249 LET PC=(NC/NP)*100
250 LET PF=(NF/NP)*100
251 REM
252 REM ***  IMPRESION DE RESULTADOS ***
253 REM
254 CLS
255 PRINT TAB(12);"RESULTADOS FINALES"
256 PRINT TAB(11);"===================="
257 PRINT
258 PRINT "NUMERO DE PREGUNTAS  = ";NP
259 PRINT "NUMERO DE ACIERTOS   = ";NC
260 PRINT "NUMERO DE ERRORES    = ";NF
261 PRINT "PORCENTAJE ACIERTOS  = ";PC;"%"
262 PRINT "PORCENTAJE FALLOS    = ";PF;"%"
263 PRINT
264 PRINT
265 PRINT
266 IF PC<40 THEN PRINT "TE QUEDA MUCHO POR APRENDER"
267 IF PC>40 AND PC<70 THEN PRINT "TIENES QUE ESTUDIAR MAS"
268 IF PC>70 AND PC<90 THEN PRINT "NO ESTA MAL PERO ..."
269 IF PC>=90 THEN PRINT "SE VE QUE ERES UN EXPERTO!"
270 PRINT
271 PRINT
272 INPUT "(QUIERES LA LISTA DE LAS PALBRAS? (S/N)";R$
273 IF R$="N" GOTO 297
274 IF R$<>"S" GOTO 272
275 REM
276 REM *** IMPRESION DE LAS PALABRAS USADAS ***
277 REM
278 LET N1=1
279 LET N2=15
280 CLS
281 PRINT "PALABRAS"
282 PRINT
283 PRINT "EN ESPA%OL","EN INGLES";"   CORRECTO"
284 PRINT "----------","----------";"   --------"
285 IF NP<N2 THEN LET N2=NP
286 FOR I=N1 TO N2
287     PRINT S$(I),I$(I)
288     LOCATE I+4,30
289     IF C(I)=1 THEN PRINT "SI":GOTO 291
290     PRINT "NO"
291 NEXT I
292 IF NP<>N2 THEN N2=N2+15
293 LET N1=N1+15
294 PRINT
295 INPUT "(CONTINUAMOS? (ENTER)";A$
```

```
296 IF NP<>N2 GOTO 280
297 PRINT
298 PRINT
299 INPUT "(QUIERES CONTINUAR CON OTRO NIVEL? (S/N)";R$
300 IF R$="S" THEN GOTO 144
301 IF R$="N" THEN END
302 GOTO 299
303 REM
304 REM
305 REM *** SIN PALABRAS ***
306 REM
307 CLS
308 PRINT
309 PRINT
310 PRINT
311 PRINT
312 PRINT "NO HAY MAS PALABRAS DEL NIVEL ";NN;"."
313 FOR I = 1 TO 5000
314 NEXT I
315 GOTO 242
316 REM
317 REM *******************************
318 REM *            DATA             *
319 REM *******************************
320 REM
321 REM ***  NIVEL 1 ***
322 REM
323 DATA "LLUVIA","RAIN","GATO","CAT","HOMBRE","MAN","HORA","HOUR","SOL","SUN","
HERMANO","BROTHER","MADRE","MOTHER","PERRO","DOG","MESA","TABLE","MINUTO","MINUT
E","PADRE","FATHER"
324 DATA "MUJER","WOMAN","HERMANA","SISTER","NIEVE","SNOW","TIO","UNCLE","TIA","
AUNT","AMIGO","FRIEND","SILLA","CHAIR"
325 DATA "POLLO","CHICKEN","COMIDA","FOOD","COCHE","CAR","NI%O","CHILD","PRIMO",
"COUISIN","CASA","HOME"
326 DATA "PUERTA","DOOR","VENTANA","WINDOW","TU","YOU","UNO","ONE","DOS","TWO","
TRES","THREE","CUATRO","FOUR","CINCO","FIVE","SEIS","SIX","SIETE","SEVEN","OCHO"
,"EIGHT","NUEVE","NINE","DIEZ","TEN"
327 DATA "LUNES","MONDAY"
328 DATA "MARTES","TUESDAY"
329 DATA "MIERCOLES","WEDNESDAY"
330 DATA "JUEVES","THURSDAY"
331 DATA "VIERNES","FRIDAY"
332 DATA "SABADO","SATURDAY"
333 DATA "DOMINGO","SUNDAY"
334 DATA "POBRE","POOR"
335 DATA "RICO","RICH"
336 DATA "DINERO","MONEY"
337 DATA "CARTA","LETTER"
338 DATA "PALABRA","WORD"
339 DATA "NOCHE","NIGHT"
340 DATA "LUNA","MOON"
341 REM
342 REM  ***  NIVEL 2 ***
343 REM
344 DATA "RATON","MOUSE"
345 DATA "MONO","MONKEY"
346 DATA "MES","MONTH"
347 DATA "A%O","YEAR"
348 DATA "BOCA","MOUTH"
349 DATA "MOMENTO","MOMENT"
350 DATA "MEDIANOCHE","MIDNIGHT"
351 DATA "CARNE","MEAT"
352 DATA "LECHE","MILK"
353 DATA "HUEVOS","EGGS"
354 DATA "AGUA","WATER"
355 DATA "HIELO","ICE"
356 DATA "LIBRE","FREE"
357 DATA "LECHUGA","LETTUCE"
358 DATA "TOMATE","TOMATO"
359 DATA "CEBOLLA","ONION"
360 DATA "ENERO","JANUARY"
361 DATA "FEBRERO","FEBRUARY"
362 DATA "MARZO","MARCH"
363 DATA "ABRIL","APRIL"
364 DATA "MAYO","MAY"
365 DATA "JUNIO","JUNE"
366 DATA "JULIO","JULY"
367 DATA "AGOSTO","AUGUST"
368 DATA "SEPTIEMBRE","SEPTEMBER"
369 DATA "OCTUBRE","OCTOBER"
370 DATA "NOVIEMBRE","NOVEMBER"
371 DATA "DICIEMBRE","DECEMBER"
372 DATA "FECHA","DATE"
373 DATA "TIEMPO","WEATHER"
374 DATA "OTO%O","AUTUMN"
```

```
375 DATA "VERANO","SUMMER"
376 DATA "INVIERNO","WINTER"
377 DATA "PRIMAVERA","SPRING"
378 DATA "ESTRELLA","STAR"
379 DATA "DERECHO","RIGHT"
380 DATA "IZQUIERDO","LEFT"
381 DATA "CAMINO","ROAD"
382 DATA "CALLE","STREET"
383 DATA "AZUL","BLUE"
384 DATA "ROJO","RED"
385 DATA "VERDE","GREEN"
386 DATA "NARANJA","ORANGE"
387 DATA "AMARILLO","YELLOW"
388 DATA "BLANCO","WHITE"
389 DATA "NEGRO","BLACK"
390 DATA "PLATANO","BANANA"
391 DATA "ACEITUNA","OLIVE"
392 DATA "AHORA","NOW"
393 DATA "DONDE","WHERE"
394 DATA "NOMBRE",NAME"
395 DATA "LAGO","LAKE"
396 REM
397 REM  ***  NIVEL 3 ***
398 REM
399 DATA "TINTA","INK"
400 DATA "AQUI","HERE"
401 DATA "FRIO","COLD"
402 DATA "CALOR","WARM"
403 DATA "CALIENTE","HOT"
404 DATA "SUELO","FLOOR"
405 DATA "NIEBLA","FOG"
406 DATA "TENEDOR","FORK"
407 DATA "CUCHILLO","KNIFE"
408 DATA "CUCHARA","SPOON"
409 DATA "DEPORTE","SPORT"
410 DATA "SUR","SOUTH"
411 DATA "NORTE","NORTH"
412 DATA "OESTE","WEST"
413 DATA "ESTE","EAST"
414 DATA "FACIL","EASY"
415 DATA "OREJA","EAR"
416 DATA "CODO","ELBOW"
417 DATA "EMPLEADO","EMPLOYEE"
418 DATA "TRABAJO","JOB"
419 DATA "OFICINA","OFFICE"
420 DATA "EDIFICIO","BUILDING"
421 DATA "MUSICA","MUSIC"
422 DATA "DESAYUNO","BREAKFAST"
423 DATA "DICCIONARIO","DICTIONARY"
424 DATA "DIENTE","TOOTH"
425 DATA "DIOS","GOD"
426 DATA "ENSALADA","SALAD"
427 DATA "ESCLAVO","SLAVE"
428 DATA "FISICA","PHYSICS"
429 DATA "FRASE","SENTENCE"
430 DATA "MEDICINA","MEDICINE"
431 DATA "MIEDO","FEAR"
432 DATA "CANSADO","TIRED"
433 DATA "ENFERMO","SICK"
434 DATA "FEO","UGLY"
435 DATA "SUCIO","DIRTY"
436 DATA "LIMPIO","CLEAN"
437 DATA "VERBO","VERB"
438 DATA "VOZ","VOICE"
439 DATA "VOCAL","VOWEL"
440 DATA "PATATA","POTATO"
441 DATA "CAMA","BED"
442 DATA "FARMACIA","PHARMACY"
443 DATA "PLATO","PLATE"
444 DATA "BOLSA","PURSE"
445 DATA "BOLSILLO","POCKET"
446 DATA "MILITAR","MILITARY"
447 DATA "PRESIDENTE","PRESIDENT"
448 DATA "PAIS","COUNTRY"
449 DATA "VACA","COW"
450 DATA "COCINA","KITCHEN"
451 REM
452 REM  ***   NIVEL 5 ***
453 REM
454 DATA "ABOGADO","ATTORNEY"
455 DATA "BIBLIOTECA","LIBRARY"
456 DATA "ESCUELA","SCHOOL"
457 DATA "VIDA","LIFE"
```

```
458 DATA "ESPEJO","MIRROR"
459 DATA "MILAGRO",'MIRACLE"
460 DATA "CARTERO","POSTMAN"
461 DATA "POBLACION","POPULATION"
462 DATA "ARROZ","RICE"
463 DATA "GOMA","RUBBER"
464 DATA "RIESGO","RISK"
465 DATA "ESTILO","STYLE"
466 DATA "FUERZA","STRENGTH"
467 DATA "CORRIENTE","STREAM"
468 DATA "VELOCIDAD","SPEED"
469 DATA "RUTINA","ROUTINE"
470 DATA "POSDATA","POSTSCRIPT"
471 DATA "HUELLA","NECK"
472 DATA "TIERRA","LAND"
473 DATA "RANA","FROG"
474 DATA "EMPERADOR","EMPEROR"
475 DATA "COCINERO","COOK"
476 DATA "PELICULA","MOVIE"
477 DATA "OPORTUNIDAD","CHANCE"
478 DATA "MEJILLA","CHEEK"
479 DATA "CABALLERO","GENTLEMAN"
480 DATA "CUERNO","HORN"
481 DATA "MENTIROSO","LIAR"
482 DATA "LUZ","LIGHT"
483 DATA "HORNO","OVEN"
484 DATA "PREMIO","PRIZE"
485 DATA "ESPACIO","SPACE"
486 DATA "ESTADO","STATE"
487 DATA "CUADRO","SQUARE"
488 DATA "EQUIPO","TEAM"
489 DATA "IMPUESTO","TAX"
490 DATA "CINTA","TAPE"
491 DATA "GARGANTA","THROAT"
492 DATA "JUGUETE","TOY"
493 DATA "LENGUA","TONGUE"
494 DATA "BARCA","BOAT"
495 DATA "BARRIO","NEIGHBORHOOD"
496 DATA "CAMISA","SHIRT"
497 DATA "CARTEL","POSTER"
498 DATA "CLIMA","CLIMATE"
499 DATA "FUEGO","FIRE"
500 DATA "HAMBRE","HUNGER"
501 DATA "ISLA","ISLAND"
502 DATA "JEFE","CHIEF"
503 DATA "LAGRIMA","TEAR"
504 DATA "LLAVE","KEY"
505 DATA "OLOR","SMELL"
```





Aunque es un programa muy largo, merece la pena intentar introducirlo, ya que los resultados son mejores de lo que uno espera.

El programa puede funcionar perfectamente, y sin problemas, en cualquier tipo de IBM o compatible así como en el AMSTRAD. Para el resto de los ordenadores las modificaciones son las siguientes:

COMMODORE:

```
132 PRINT CHR$(147)
138 POKE 214,8:POKE 211,12:PRINT "EN-
```



Ejemplo de ejecución del programa «Vocabulario inglés».

```
GLISH VOCABULARY"
139 POKE 214,9:POKE 211,11:PRINT
"-------------------"
140 POKE 214,12:POKE 211,6:PRINT "(c) Ed.
Siglo Cultural, 1987"
147 POKE 214,17:POKE 211,2
150 A=RND(TI)
168 LET RN=INT(RND(1)*50+1)
174 PRINT CHR$(147):PRINT:PRINT
181 POKE 214,11:POKE 211,INT((40-
NL)/2)-1
185 POKE 214,10:POKE
211,INT((40/NL)/2)-3
187 PRINT CHR$(147)
203 POKE 214,12:POKE 211,14
207 POKE 214,12:POKE 211,14
212 POKE 214,15:POKE 211,0
224 POKE 214,12:POKE 211,7
228 POKE 214,12:POKE 211,7
233 POKE 214,13:POKE 211,0
237 POKE 214,15:POKE 211,0
254 PRINT CHR$(147)
280 PRINT CHR$(147)
288 POKE 214,I+4:POKE 211,29
307 PRINT CHR$(147)
```

MSX:

```
138 LOCATE 13,8:PRINT "ENGLISH VOCABU-
LARY"
139 LOCATE 12,9:PRINT "-------------------"
140 LOCATE 7,12:PRINT "(c) Ed. Siglo Cul-
tural, 1987"
147 LOCATE 3,17
150 RANDOMIZE TIME
168 LET RN=INT(RND(1)*50+1)
181 LOCATE INT((40-NL)/2),11
185 LOCATE INT((40-NL)/2)-2,10
203 LOCATE 15,12
207 LOCATE 15,12
212 LOCATE 1,15
224 LOCATE 8,12
228 LOCATE 8,12
233 LOCATE 1,13
237 LOCATE 1,15
288 LOCATE 30,I+4
```

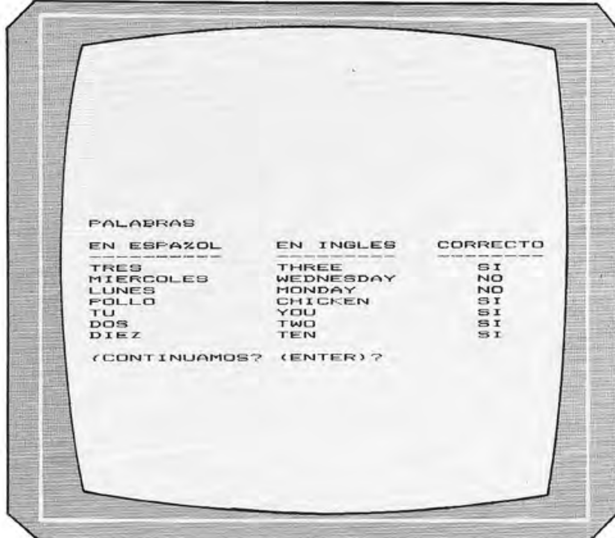
SPECTRUM:

```
133 PRINT "********************************"
135 PRINT "*";TAB(31);"*"
137 PRINT "********************************"
138 PRINT AT 8,9;"ENGLISH VOCABULARY"
139 PRINT AT 9,8;"-------------------"
140 PRINT AT 12,3;"(c) Ed. Siglo Cultural,
1987"
147 REM
150 RANDOMIZE 0
181 PRINT AT 11,INT(32-NL)/2);
185 PRINT AT 10,INT(32-NL)/2)-2;
186 INPUT E$:PRINT E$
203 PRINT AT 12,11;
207 PRINT AT 12,11;
212 PRINT AT 15,1;
224 PRINT AT 12,4;
228 PRINT AT 12,4;
233 PRINT AT 13,1;
237 PRINT AT 15,1;
288 PRINT AT I+4,26;
```





El programa da el tanto por ciento de aciertos y de errores.





Listas de las palabras que se le preguntaron al usuario.



Notas sobre el programa 2

Este programa está especialmente diseñado con una idea de crecimiento continuo. Esto significa que está pensado para que el usuario en cualquier momento pueda variar el número de palabras por nivel e incluso de niveles. Veamos cómo hacerlo.

Si queremos tener 100 palabras por nivel, en vez de las 50 que hay actualmente, sólo tendremos que cambiar las siguientes líneas:

```
127 DIM S$(100):DIM I$(100)

128 DIM C(100):DIM U(100)

152 FOR I=1 TO 100

160 IF NP=101 THEN GOTO 304

168 LET RN=INT(RND*100+1)
```

introduciendo, lógicamente, 50 palabras para cada nivel, en las líneas DATA.



Programa: Calendarios

Este programa nos permitirá imprimir calendarios por la impresora. Para ello es necesario que el usuario tenga una impresora de 80 columnas como mínimo. Por ello, este programa no funciona en el SPECTRUM, aunque en tomos sucesivos aparecerá la versión para SPECTRUM.





Pantalla de presentación del programa «Calendario».

```
CALENDARIOS
===========

10 REM*****************************************************************************
11 REM*                                                                           *
12 REM*  CCCC   AAAA  L      EEEEEE N    N DDDD   AAAA  RRRR  IIIII  OOO   SSS  *
13 REM* C    C A    A L      E    E NN   N D    D A    A R    R   I   O   O S   S *
14 REM* C      A    A L      E  E   N N N  D    D A    A R    R   I   O   O S     *
15 REM* C      AAAAAA L      EEEE   N N N  D    D AAAAAA RRRR     I   O   O  SSS  *
16 REM* C      A    A L      E  E   N N N  D    D A    A R R      I   O   O     S *
17 REM* C    C A    A L      E    E N   NN D    D A    A R   R    I   O   O S   S *
18 REM*  CCCC  A    A LLLLLL EEEEEE N    N DDDD   A    A R    R IIIII  OOO   SSS  *
19 REM*                                                                           *
20 REM*****************************************************************************
21 REM
22 REM    ****************************************************************
23 REM    ****************************************************************
24 REM    *************** PROGRAMADO POR FRANCISCO MORALES ***************
25 REM    ****************************************************************
26 REM
27 REM    ****************************************************************
28 REM    ****************************************************************
29 REM    ***************** (c) Ed. Siglo Cultural, 1987 *****************
30 REM    ****************************************************************
31 REM
100 CLS
```

```
101 PRINT "   ******************************************************************
********"
102 PRINT "   *
       *"
103 PRINT "   *  ####   ####  #      ###### #   # ####   ####  ####  #####  ### 
  ###  *"
104 PRINT "   * #    # #    # #      #    # ##  # #   # #    # #   #   #   #   #
 #   # *"
105 PRINT "   * #      #    # #      #  #   # # # #   # #    # #   #   #   #   #
 #     *"
106 PRINT "   * #      ###### #      ####   # # # #   # ###### ####    #   #   #
  ###  *"
107 PRINT "   * #      #    # #      #  #   # # # #   # #    # # #    #   #   #
     # *"
108 PRINT "   * #    # #    # #      #    # #  ## #   # #    # #  #   #   #   #
 #   # *"
109 PRINT "   *  ####  #    # ###### ###### #   # ####  #    # #   # #####  ###
  ###  *"
110 PRINT "   *
       *"
111 PRINT "   ******************************************************************
********"
112 PRINT
113 PRINT
114 PRINT "   (c) Ed. Siglo Cultural."
115 FOR I=1 TO 1000
116 NEXT I
117 PRINT:PRINT
118 PRINT "   AJUSTA EL PAPEL DE LA IMPRESORA."
119 PRINT
120 PRINT "   ENCIENDE LA IMPRESORA."
121 PRINT
122 PRINT "   PULSA UNA TECLA CUANDO ESTES LISTO"
123 LET A$=INKEY$:IF A$="" THEN GOTO 123
124 LET M$="1987"
125 DIM M(12)
126 LET D=-3
127 LET S=0
128 FOR N=0 TO 12
129    READ M(N)
130 NEXT N
131 LPRINT "  CALENDARIO DE ";M$
132 LPRINT "----------------------"
133 LPRINT:LPRINT
134 FOR N=1 TO 12
135    CLS
136    PRINT "IMPRIMIENDO EL CALENDARIO DEL A%O ";M$
137    PRINT:PRINT
138    PRINT "ESTOY IMPRIMIENDO EL MES nO. ";N
139    PRINT:PRINT:PRINT
140    LPRINT:LPRINT:LPRINT:LPRINT:LPRINT:LPRINT
141    LET S=S+M(N-1)
142    PRINT "**";S;TAB(8);
143    LPRINT "**";S;TAB(8);
144    FOR I=1 TO 17
145       PRINT "*";
146       LPRINT "*";
147    NEXT I
148    ON N GOTO 149,150,151,152,153,154,155,156,157,158,159,160
149    PRINT "  ENERO    ";:LPRINT "  ENERO    ";:GOTO 161
150    PRINT " FEBRERO   ";:LPRINT " FEBRERO   ";:GOTO 161
151    PRINT "  MARZO    ";:LPRINT "  MARZO    ";:GOTO 161
152    PRINT "  ABRIL    ";:LPRINT "  ABRIL    ";:GOTO 161
153    PRINT "  MAYO     ";:LPRINT "   MAYO    ";:GOTO 161
154    PRINT "  JUNIO    ";:LPRINT "  JUNIO    ";:GOTO 161
155    PRINT "  JULIO    ";:LPRINT "  JULIO    ";:GOTO 161
156    PRINT " AGOSTO    ";:LPRINT "  AGOSTO   ";:GOTO 161
157    PRINT "SEPTIEMBRE";:LPRINT "SEPTIEMBRE";:GOTO 161
158    PRINT " OCTUBRE   ";:LPRINT " OCTUBRE   ";:GOTO 161
159    PRINT "NOVIEMBRE ";:LPRINT "NOVIEMBRE ";:GOTO 161
160    PRINT "DICIEMBRE ";:LPRINT "DICIEMBRE ";
161    FOR I=1 TO 18
162       PRINT "*";
163       LPRINT "*";
164    NEXT I
165    PRINT 365-S;"**"
166    LPRINT 365-S;"**"
167    PRINT:PRINT "     L       M       X       J       V       S       D"
168    LPRINT:LPRINT "     L       M       X       J       V       S       D"
169    PRINT
170    LPRINT
171    FOR I=1 TO 59
172       PRINT "*";
```

```
173         LPRINT "*";
174      NEXT I
175      FOR W=1 TO 6
176         PRINT
177         LPRINT
178         PRINT TAB(4);
179         LPRINT TAB(4);
180         FOR G=1 TO 7
181            LET  D=D+1
182            LET D2=D-S
183            IF D2>M(N) THEN 190
184            IF D2>0 THEN PRINT D2;:LPRINT D2;
185            PRINT TAB(4+8*G);
186         LPRINT TAB(4+8*G);
187         NEXT G
188         IF D2=M(N) THEN 196
189      NEXT W
190      LET D=D-G
191 LPRINT:LPRINT
192      FOR I=1 TO 59
193         PRINT "*";
194         LPRINT "*";
195      NEXT I
196 NEXT N
197 FOR I=1 TO 6
198      PRINT
199 NEXT I
200 DATA 0,31,28,31,30,31,30,31,31,30,31,30,31
```

El programa puede funcionar sin cambios en el IBM, MSX y en el AMSTRAD. Para el COMMODORE los cambios que hay que hacer son:

COMMODORE:

```
100 PRINT CHR$(147)
131 PRINT "CALENDARIO DE 1987"
132 PRINT "-----------------"
133 PRINT:PRINT
135 REM
136 REM
137 REM
138 REM
138 REM
140 PRINT:PRINT:PRINT:PRINT:PRINT:PRINT
```

También hay que quitar todos los LPRINT del listado y poner en la línea 133 PRINT:PRINT:OPEN 1,4:CMD.



Notas sobre el programa 3

El programa está preparado para sacarnos el calendario de 1987. Si queremos que nos saque otro año tenemos que hacer lo siguiente.

1. Poner el año en la línea 124 tal y como aparece en el listado original.
2. Asignarle a la variable numérica D, que se encuentra en la línea 126, un número que depende del día de la semana con que empiece el año. Si el año empieza en lunes pondremos un 0. Si empieza en martes un −1 y así sucesivamente.

— LUNES = 0
— MARTES = −1
— MIERCOLES = −2
— JUEVES = −3
— VIERNES = −4
— SABADO = −5
— DOMINGO = −6

3. Si el año resulta ser bisiesto entonces hay que cambiar las líneas siguientes:

```
165 PRINT 366-S
166 LPRINT 366-S
200 DATA 0,31,29,31,30,31,30,31,31,30,
31,30,31
```



Programa: Cuatro en raya

Este programa sólo funciona en el ordenador IBM PC, XT, AT y compatibles. Los que tengáis otro ordenador no os preocupéis, este mismo programa aparecerá en tomos sucesivos. El programa es el número 2.

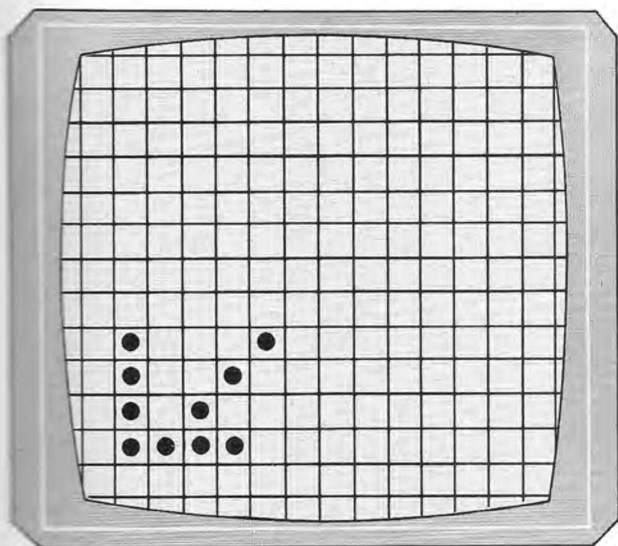
```
CUATRO EN RAYA
==============

10 REM **************************************************************
11 REM *                                                            *
12 REM *       444                                                  *
13 REM *      4444                                                  *
14 REM *     44 44                                                  *
15 REM *    44  44      EEEEE N    N   RRRR   AAA  Y   Y   AAA      *
16 REM *   444444444    E     NN   N   R   R A   A Y   Y  A   A     *
17 REM *        44      EEE   N N N    RRRRR AAAAA  Y Y   AAAAA     *
18 REM *        44      E     N  NN    R  R  A   A   Y    A   A     *
19 REM *       4444     EEEEE N    N   R   R A   A  .Y    A   A     *
20 REM *                                                            *
21 REM **************************************************************
22 REM *                                                            *
23 REM *  PROGRAMA REALIZADO POR Carlos Coral y Fco. Morales.       *
24 REM *                                                            *
25 REM *  (c) EDICIONES SIGLO CULTURAL, 1987                        *
26 REM *                                                            *
27 REM **************************************************************
28 REM
100 DIM S$(2)
101 LET S$(1)="*"
102 LET S$(2)="#"
103 SCREEN 0
104 CLS
105 PRINT "4 EN RAYA"
106 PRINT "----------"
107 PRINT
108 PRINT "  El juego consiste en conseguir que"
109 PRINT "cuatro de nuestras fichas se encuentren"
110 PRINT "en linea. Esta linea puede ser vertical,"
111 PRINT "horizontal y diagonal."
112 PRINT
113 PRINT "  Cuando el ordenador pregunte cual es"
114 PRINT "tu movimiento tienes que darle la fila y"
115 PRINT "la columna en la que te quieres colocar."
116 PRINT "Un ejemplo. Si te vas a mover a la colum"
117 PRINT "na H de la fila 4, tendras que escribir"
118 PRINT "H4."
119 PRINT
120 PRINT "  El jugador No. 1 tiene la ficha *."
121 PRINT
122 PRINT "  El jugador No. 2 tiene la ficha #."
123 PRINT
124 PRINT "PULSA UNA TECLA Y ..."
125 PRINT "....BUENA SUERTE !!!"
126 IF INKEY$="" THEN GOTO 126
127 CLS
128 LOCATE 1,3
129 PRINT "1  2  3  4  5  6  7  8  9 10 11 12"
130 LET R=65
131 FOR F=3 TO 17 STEP 2
132    LOCATE F,1
133    PRINT CHR$(R)
134    LET R=R+1
135 NEXT F
136 FOR F=3 TO 17 STEP 2
137    LOCATE F,3
138    PRINT ".  .  .  .  .  .  .  .  .  .  .  ."
139 NEXT F
140 LOCATE 20,1
141 PRINT "PULSA UNA TECLA"
142 IF INKEY$="" THEN GOTO 140
143 LOCATE 20,1
144 PRINT "                                        "
145 LOCATE 20,1
146 INPUT "NOMBRE DE JUGADOR 1= ";A$
147 LOCATE 20,1
148 PRINT "                                        "
149 LOCATE 20,1
150 INPUT "NOMBRE DE JUGADOR 2= ";B$
151 LOCATE 20,1
152 PRINT "                                        "
```

```
153 LOCATE 20,1
154 INPUT "¿ QUIEN VA A EMPEZAR (1,2) ";TR
155 IF TR<1 OR TR>2 THEN GOTO 153
156 DIM A(135)
157 IF TR=1 THEN LET C$=A$:GOTO 159
158 LET C$=B$
159 LOCATE 20,1
160 PRINT "
         "
161 LOCATE 20,1
162 PRINT "TURNO DEL JUGADOR";TR;" (";S$(TR);")"
163 INPUT "¿DONDE VAS A MOVER? (COL.FILA) ";R$
164 IF R$="" OR LEN(R$)>3 THEN GOTO 159
165 LET J$=MID$(R$,1,1):IF J$<"A" OR J$>"H" THEN GOTO 159
166 LET FI=ASC(J$)-64
167 LET J$=MID$(R$,2,3):IF VAL(J$)<1 OR VAL(J$)>12 THEN GOTO 159
168 LET CO=VAL(J$)
169 LET SI=((FI-1)*12)+CO
170 IF A(SI)<>0 THEN GOSUB 207:GOTO 159
171 GOSUB 201
172 IF ER=1 THEN 159
173 FOR J=3 TO 1+FI*2 STEP 2
174    LOCATE J,3+((CO-1)*3)
175    PRINT S$(TR)
176    FOR K=1 TO 100
177    NEXT K
178    LOCATE J,3+((CO-1)*3)
179    PRINT "."
180 NEXT J
181 LOCATE 1+FI*2,3+((CO-1)*3)
182 PRINT S$(TR)
183 LET A(SI)=TR
184 LOCATE 20,1
185 PRINT "
         ";
186 GOSUB 189
187 IF TR=1 THEN LET TR=2:GOTO 158
188 LET TR=1:GOTO 157
189 LET E=1
190 LET A=12
191 FOR F=E TO A
192    IF A(F)=TR AND A(F+1)=TR AND A(F+2)=TR AND A(F+3)=TR THEN GOTO 214
193    IF A(F)=TR AND A(F+13)=TR AND A(F+26)=TR AND A(F+39)=TR THEN GOTO 214
194    IF A(F)=TR AND A(F+11)=TR AND A(F+22)=TR AND A(F+33)=TR THEN GOTO 214
195    IF A(F)=TR AND A(F+12)=TR AND A(F+24)=TR AND A(F+36)=TR THEN GOTO 214
196 NEXT F
197 LET E=E+12
198 LET A=E+11
199 IF A<108 THEN GOTO 191
200 RETURN
201 FOR F=FI TO 7
202 LET SP=SI+12
203 IF A(SP)=0 THEN GOSUB 207:LET ER=1:RETURN
204 NEXT F
205 LET ER=0
206 RETURN
207 LOCATE 20,1
208 PRINT "
         "
209 LOCATE 20,1
210 PRINT "ESA JUGADA NO ES VALIDA"
211 FOR X=1 TO 2000
212 NEXT X
213 RETURN
214 LOCATE 20,1
215 PRINT "4 EN RAYA"
216 FOR F=1 TO 100
217    SOUND 100+INT(RND*800),.5
218 NEXT F
219 LOCATE 21,1
220 PRINT "
221 LOCATE 20,1
222 PRINT "HAS GANADO ";C$
223 LOCATE 21,1
224 PRINT "PULSA UNA TECLA"
225 IF INKEY$="" THEN GOTO 225
226 LOCATE 20,1
227 PRINT "
         "
228 LOCATE 20,1
229 INPUT "QUIERES JUGAR OTRA PARTIDA (S/N)";A$
230 IF A$="S" OR A$="s" THEN RUN
```

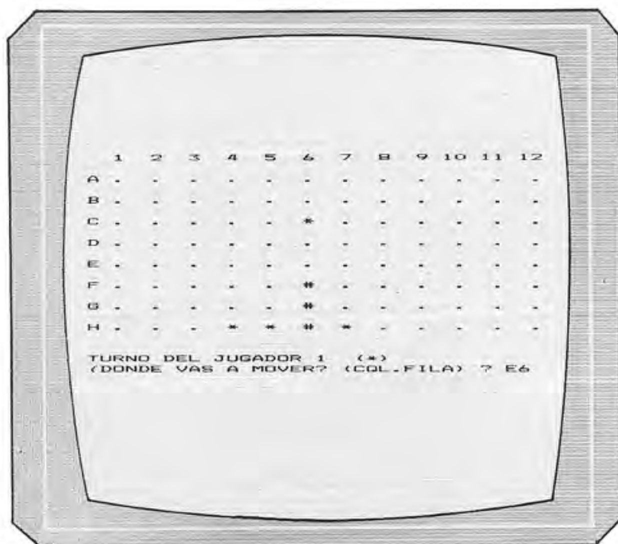
```
231 IF A$<>"N" AND A$<>"n" THEN GOTO 226
232 CLS
233 PRINT "A D I O S"
234 PRINT "========="
235 END
```

Este programa simula el juego de las cuatro en raya que hay en muchas casas. El objetivo de este juego es conseguir que cuatro de nuestras fichas estén en línea recta antes de que lo consiga nuestro contrincante. Aparte de lo difícil que pueda resultar esto, se plantea otro problema. El tablero sobre el cual jugaremos, está puesto verticalmente de manera que no se puede poner una ficha en una fila si en la fila anterior no hay otra. Con esto se consigue que no podamos poner nuestra ficha en donde nosotros queramos.



Estas son las tres maneras posibles de hacer 4 en raya, vertical, horizontal y diagonalmente.

Este juego está planteado para dos jugadores, aunque el ordenador hace de árbitro e indica quién es el jugador. A cada jugador le corresponde una ficha diferente y cada jugador tiene un turno para jugar. En este juego no se pueden variar las jugadas, por lo que es necesario pensar mucho antes de poner nuestra ficha.





Ejemplo de ejecución del programa 4 en raya.



Notas sobre el programa 4

Cuando el ordenador te pida tu jugada tendrás que indicar la fila y la columna en la que te quieres posicionar, en este orden. Según esto, si te quieres mover a la columna 5 de la fila H, tendrás que darle dicho movimiento a tu ordenador como H5.

En el caso de que pusiese 5H en vez de H5, el ordenador no te hará caso y te volverá a preguntar. Puedes intentar, si quieres, que el ordenador acepte cualquiera de las dos formas.

Este programa aparecerá para el resto de los ordenadores en el tomo 8.



Programa: Copy de pantalla para SPECTRUM

Este quinto programa que vamos a ver a continuación sólo es válido para los

usuarios del SPECTRUM. Con él podréis hacer que cualquier pantalla, de cualquier juego, que os guste, se grabe en una cinta de cassette para, más tarde, poderla cargar para admirarla o experimentar con ella.

Esta rutina es realmente útil para todos aquellos que les guste almacenar, ya sean sus propias pantallas o las de los juegos comerciales, en cintas de cassette.

A continuación os damos el listado en ensamblador para todos aquellos que quieran ver cómo funciona.

```
 10 ;******************************
 20 ;* COPY DE PANTALLA MEDIANTE *
 30 ;*        INTERRUPCIONES      *
 40 ;******************************
 50 ;
 60 ;******************************
 70 ;* PROGRAMADO POR:            *
 80 ;*                            *
 90 ;*        CARLOS CORAL        *
 95 ;******************************
 96 ;
100         ORG 65279
110         DEFW RUTINA
120         ORG 65200
130         LD A,254
140         LD I,A
150         IM 2
160         RET
170 ;
172 ;**********************
174 ;
180 RUTINA DI
190         PUSH AF
200         PUSH BC
210         PUSH DE
220         PUSH IX
230         PUSH IY
240         LD A,127
250         IN A,(254)
260         BIT 1,A
270         CALL Z,TECLA2
280         RST 56
290         POP IY
300         POP IX
310         POP HL
320         POP DE
330         POP BC
340         POP AF
350         RETI
360 ;
370 ;**********************
380 ;
390 TECLA2 LD A,191
400         IN A,(254)
410         BIT 0,A
420         RET NZ
430         SCF
440         LD A,255
450         LD DE,6912
460         LD IX,16384
470         CALL 1218
480         RET
```

Para todos los que estén interesados en esta rutina damos el listado del programa en BASIC que se encarga de almacenar este programa en CODIGO MAQUINA en la memoria.

```
 COPY DE PANTALLA
-------------------

  10 REM ***********************
  20 REM * COPY DE PANTALLA ME-*
  30 REM *DIANTE INTERRUPCIONES*
  40 REM ***********************
  50 REM
  60 PAPER 0: BORDER 0: INK 6: C
LEAR 65199
  70 PRINT "--------------------
-------------        COPY DE LA PA
NTALLA      ---------------------
-------------"
  80 PRINT #0; FLASH 1;"         L
EYENDO LINEAS DATA          "
  90 PRINT AT 10,7;"ESPERA UN MO
MENTO"
 100 PRINT AT 12,8;"CHECKSUM = "
;
 105 LET SUMA=0
 110 RESTORE 5000
 120 FOR I=0 TO 56
 130    READ A
 140    LET SUMA=SUMA+A
 150    POKE 65200+I,A
 160 PRINT AT 12,19;SUMA
 170  NEXT I
 175 POKE 65279,133: POKE 65280,
254
 180 IF SUMA=9720 THEN  GO TO 30
0
 190 PRINT AT 14,13; FLASH 1;"ER
ROR"
 200 BEEP 2,-10
 210 CLS
 220 PRINT " Hay algun error en
las lineas  DATA."''" Repasalas
antes de continuar"
 230 GO TO 9999
 300 CLS
 310 PRINT AT 1,8; FLASH 1;"RUTI
NA INSTALADA"
 320 PRINT '''" Para utilizarla
solo tienes quepulsar las teclas
 symbol-shift yENTER a la vez"
 330 RANDOMIZE USR 65200
1000 PAUSE 0
4900  REM *******************
4910  REM ** LINEAS DE DATA **
4920 REM *******************
4930 REM
5000 DATA 62,254,237,71,237,94,2
01,243,245,197
5010 DATA 213,229,221,229,253,22
9,62,127,219,254
5020 DATA 203,79,204,212,254,255
,253,225,221,225
5030 DATA 225,209,193,241,237,77
,62,191,219
5040 DATA 254,203,71,192,55,62,2
55,17,0,27,221
5050 DATA 33,0,64,205,194,4,201
```

Para utilizar esta rutina sólo necesitas hacer RUN al programa N.º 2. Si has escrito bien todos los números de las líneas DATA, te aparecerá un mensaje en la pantalla comunicándote que el programa está almacenado en la memoria.

Un vez que tenemos el programa en la memoria, para hacer que se ejecute, sólo tenemos que pulsar las teclas:

SYMBOL-SHIFT y ENTER

a la vez. Haciendo esto se grabará inmediatamente la pantalla que estemos viendo en ese momento en el ordenador.



Notas sobre los programas 5 y 6

Si tienes un ensamblador de Z80 y no te gusta en qué dirección de memoria hemos colocado la rutina, puedes ponerla en otro lugar de la memoria y utilizarla en tus propios programas.

La rutina no espera que pulses ninguna tecla más después de pulsar SYMBOL-SHIFT y ENTER. Por ello es conveniente que tengas la cinta y el cassette preparado antes de pulsarlas.

El programa que acabamos de ver graba la pantalla pero no graba ninguna cabecera. Como sabes, si no está grabada la cabecera, el SPECTRUM no puede leer la pantalla. Caben dos posibles soluciones a este problema:

1. Realizar otra rutina en CODIGO MAQUINA que se encargue de leer pantallas sin cabecera. –
2. Hacer lo que te decimos a continuación paso a paso:

A. Coger una cinta virgen y rebobinarla.
B. Pulsar, a la vez, las teclas RECORD y PLAY.
C. Dejar que la cinta corra durante 10 segundos.
D. Escribir en el SPECTRUM

SAVE "nombre" SCREEN$

donde **nombre** es el nombre de la pantalla que quieres grabar.

E. Pulsa la tecla ENTER dos veces.
F. Cuando el ordenador haya grabado la cabecera, para el cassette.
G. Ya tenemos grabada la cabecera que necesitábamos. Ahora sólo nos queda ejecutar el programa 4, cargar el juego del que queremos copiar la pantalla y cuando tengamos la pantalla a la vista pulsar las teclas SYMBOL-SHIFT y ENTER.

Puede que esta rutina no te funcione con algunos juegos, sobre todo si este fue realizado en los últimos dos años. Esto es debido a que muchos de los programas comerciales escritos en CODIGO MAQUINA utilizan rutinas que usan las interrupciones para realizar el movimiento de los gráficos o la música del programa. Esto no tiene una fácil solución ya que cada programa necesitaría de una rutina específica, y que sólo sirviese para 1, si queremos copiar sus pantallas.



Master Mind numérico para Spectrum

El programa que aparece a continuación, y que sólo funciona en el SPECTRUM, es el típico MASTER MIND al que todos hemos jugado más de una vez. La particularidad de este MASTER MIND computarizado es que podemos elegir la longitud de la combinación a adivinar así como el número de intentos que nos permite el ordenador.

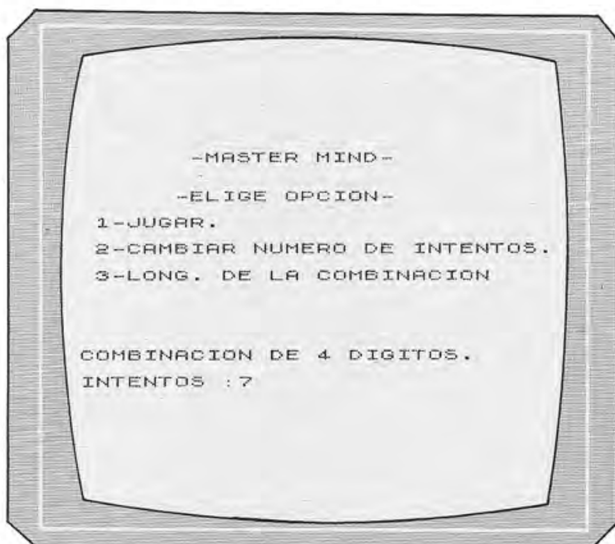
```
MASTER MIND
-------------

  10 DEF FN c$(a$,a)=CHR$ 22+CHR
$ a+CHR$ INT ((31-LEN a$)/2)+a$
  20 REM MASTER MIND
  30 REM1987 A.G.B
  40 GO SUB 1000
  50 GO SUB 1040
  60 REM COMIENZO DEL JUEGO
  70 CLS
  80 REM DIBUJA TABLERO
  90 LET CONT=1
 100 PLOT 0,151: DRAW 255,0: PLO
T 127,159: DRAW 0,-159
 110 PRINT AT 2,0;" JUGADA";AT 2
,16;"MUERTOS  HERIDOS"
 120 PRINT AT 0,2;"Jugada numero
 : ";cont
 130 REM GENERA COMBINACION
 140 LET C$=""
 150 FOR I=1 TO NNUM
 160 LET NTEMP=INT (RND*10)
 170 LET C$=C$+STR$ NTEMP
 180 NEXT I
```

```
 190 INPUT "CUAL ES TU INTENTO ?
 "; LINE I$
 200 IF LEN I$<>NNUM THEN  GO SU
B 1700: GO TO 190
 210 LET MUERTOS=0: LET HERIDOS=
0
 220 DIM G(NNUM): DIM T(NNUM)
 230 FOR I=1 TO NNUM
 240 IF C$(I)<>I$(I) THEN  GO TO
 260
 250 LET MUERTOS=MUERTOS+1: LET
G(I)=1: LET T(I)=1
 260 NEXT I
 270 FOR I=1 TO NNUM
 280 FOR G=1 TO NNUM
 290 IF I$(G)=C$(I) AND G(I)=0 A
ND T(G)=0 THEN  LET HERIDOS=HERI
DOS+1: LET T(G)=1: LET G(I)=1
 300 NEXT G
 310 NEXT I
 320 PRINT AT CONT+3,0;"     ";I$
;AT CONT+3,19;MUERTOS;AT CONT+3,
27;HERIDOS
 325 IF I$=C$ THEN  GO TO 500
 330 LET CONT=CONT+1
 340 IF CONT>NJUG THEN  GO TO 70
0
 350 PRINT AT 0,18;CONT
 360 GO TO 190
 500 REM GANO
 510 RESTORE 600
 520 READ A,B: IF A=0 THEN  GO T
O 540
 530 BEEP A,B: GO TO 520
 540 PRINT #0; FLASH 1;FN C$("-E
NHORABUENA,ADIVINASTE-",0)
 550 PAUSE 0
 555 INPUT ""
 560 GO TO 740
 600 DATA .1,1,.1,1,.1,2,.1,3,.5
,7,1,12,0,0
 700 REM PERDIO
 710 FOR I=12 TO -12 STEP -1: BE
EP .1,I: NEXT I
 720 BEEP 1,-12
 730 PRINT AT 0,0,,,, FLASH 1;FN
 C$("-LO SIENTO,OTRA VEZ SERA-",
0)
 735 PRINT FN C$("-LA COMBINACIO
N ERA "+C$+" -",1)
 740 PRINT #0; INVERSE 1; FLASH
1;FN C$("-PULSA UNA TECLA-",0)
 750 PAUSE 0
 760 GO TO 40
 999 STOP
1000 REM INICIALIZA VARIABLES Y
COLORES
1010 LET NNUM=4: LET NJUG=7
1020 BORDER 1: PAPER 1: INK 7: C
LS
1030 RETURN
1040 REM MENU
1050 CLS : PRINT FN C$("-MASTER
MIND-",0)
1060 PRINT FN C$("-ELIGE OPCION-
",3)
1070 PRINT AT 5,3;"1-JUGAR.";AT
7,3;"2-CAMBIAR NUMERO DE INTENTO
S.";AT 9,3;"3-LONG. DE LA COMBIN
ACION"
1080 PRINT AT 15,2;"COMBINACION
DE ";NNUM;" DIGITOS."
1090 PRINT AT 17,2;"INTENTOS :";
NJUG
1100 INPUT OPC
1110 IF OPC<1 OR OPC>3 THEN  GO
TO 1100
1130 IF OPC=2 THEN  GO SUB 1500
1140 IF OPC=3 THEN  GO SUB 1600
1150 IF OPC=1 THEN  RETURN
1160 GO TO 1040
1500 REM NUEVO NUMERO DE INTENTO
S
1510 INPUT "CUANTOS INTENTOS ? "
;NTEMP
1520 IF NTEMP>14 OR NTEMP<1 THEN
  GO SUB 1700: GO TO 1540
1530 LET NJUG=NTEMP
1540 RETURN
1600 REM NUEVA LONGITUD DE COMBI
NACION
1610 INPUT "DE QUE LONGITUD ? ";
NTEMP
1620 IF NTEMP>11 OR NTEMP<2 THEN
  GO SUB 1700: GO TO 1640
1630 LET NNUM=NTEMP
1640 RETURN
1700 REM ERROR
1710 BEEP 1,12: PRINT #0; FLASH
1;FN C$("-ENTRADA NO ACEPTADA-",
0): PAUSE 0
1720 RETURN
```



El juego es muy sencillo. El ordenador pensará un número entre 3 y 10 dígitos de longitud y tú tienes que adivinarlo. Para ello cuentas con una serie de intentos variables entre 2 y 13.





Este es el menú del programa «Masted Mind».

El ordenador te preguntará cuál crees tú que es la solución. Después de responder, el programa te dirá cuántos vivos y cuántos muertos acertaste. Pero, ¿qué son vivos y qué son muertos? Los vivos son aquellos dígitos que has adivinado y que además los has colocado en la misma posición en la que los tiene el ordenador. Los muertos son dígitos que también has adivinado, pero que has colocado en posición distinta de la que tiene el ordena-

dor. Por ejemplo, si el ordenador pensó el número 34661 y tú le respondiste 48866, éste te dirá que tienes un vivo y dos muertos. El vivo es el primer seis que escribiste, pues se encuentra en la misma posición que el seis que pensó el ordenador. Los dos muertos son el cuatro y el otro seis que, aunque acertaste cuáles eran, no los pusiste en su posición correcta.

Jugada numero : 6

JUGADA	MUERTOS	HERIDOS
2222	0	0
3333	1	0
4444	0	0
3555	0	1
3111	1	1



El Master Mind en plena ejecución.

Por supuesto el ordenador no te dice qué números son vivos, cuáles son muertos ni cuáles no has acertado. Justamente en esto consiste el juego, en adivinar con el menor número de pistas posibles.



Notas sobre el programa 6

Puede que el número de intentos te parezca pequeño. Si quieres que el ordenador te permita hacer más de 13 intentos sólo tienes que variar la línea 1520 y cambiarla por:

```
1520 IF NTEMP>XXXX OR NTEMP<1 THEN
     GOSUB 1700:GOTO 1540
```

donde XXXX es el número máximo de intentos que deseas tener. Por ejemplo, si quiero tener la oportunidad de realizar 100 intentos la línea 1520 nos quedaría como:

```
1520 IF NTEM>100 OR NTEMP<1 THEN GO-
     SUB 1700:GOTO 1540
```

Este programa aparecerá en tomos sucesivos para el resto de los ordenadores.

TECNICAS DE ANALISIS

EL ANALISIS DE SISTEMAS INFORMATICOS

BAJO el nombre de «análisis informático» (o simplemente «análisis») se suelen agrupar una serie muy variada de técnicas y actividades relacionadas con la racionalización y sistematización en el ámbito de la informática.

En los grandes Centros de Procesos de Datos u organizaciones informáticas, el proceso de análisis es fundamental para la adecuada organización y racionalización de los procesos. Suele haber personas dedicadas específicamente a esta tarea: los «analistas». La misión de los analistas es servir de interlocutores con los usuarios finales de los procesos que se van a desarrollar, para definir con exactitud cuáles son esos procesos y sus características. Posteriormente conciben los programas informáticos mediante los cuales se desarrollarán esos procesos y preparan la documentación que los programadores necesitan para escribir los programas correspondientes del ordenador. Además, el analista redacta el resto de los documentos que aseguran una perfecta utilización de todo el sistema: normas para las personas que preparan los datos, instrucciones para el personal de explotación, etc.

Aunque a primera vista la tarea del analista puede parecer simple, no lo es. En la fase de estudio de los procesos (en contacto con las personas que los realizan, manualmente, y con los usuarios finales) es usual encontrar un número enorme de dificultades: las personales realizan, en ocasiones, muchas tareas de un modo rutinario sin saber bien por qué las hacen (o sin tener una idea clara de qué sucederá si las hicieran de otro modo, o cómo se pueden hacer de manera distinta...); las personas que utilizarán los resultados del proceso no saben con precisión, a veces, qué datos necesitan o qué otros les serían más útiles; con suma frecuencia junto con el proceso de mecanización (o informatización) de los procesos hay que establecer cambios en los sistemas de trabajo manuales, para su racionalización o simplificación, etc.

Después, cuando se va a diseñar el proceso, hay que tener amplia experiencia para hacer que el ordenador trabaje de un modo óptimo de acuerdo con sus características y con las necesidades de los procesos a realizar (tipo de datos que se manejan, volúmenes, periodicidad, etcétera); además, hay que procurar que el sistema se diseñe de un modo tal que la programación sea lo más sencilla posible.

Por otro lado, el control de todo el proceso es complicado y hay que preparar numerosos mecanismos de comprobación de datos, contraste de unos totales con otros, puntos de «chequeo» periódicos, etc., incluso en la fase de programación y puesta a punto del sistema ha de prever el analista una serie lo más exhaustiva posible de pruebas a realizar para asegurar el perfecto funcionamiento de cada programa y del conjunto.

Para el desarrollo de todas estas actividades el analista conoce y utiliza un conjunto de técnicas específicas que aseguran el éxito de su trabajo (aparte, claro está, del conocimiento general de la máquina y otros conocimientos informáticos): se suelen plasmar los diseños en dossieres ya estructurados (con lo que se asegura la uniformidad y coherencia de la información);

ESTRUCTURA TIPICA DE UN DOSSIER DE ANALISIS DE UN PROGRAMA

1. OBJETO DE UN PROGRAMA O UNIDAD DE TRATAMIENTO (U.T.).
   - 1.1. Objeto del tratamiento. Descripción general, descripción de los resultados a obtener, etc.
   - 1.2. Límites de esta U.T. Situación de la U.T. en el conjunto del sistema global propuesto, referencias a otros documentos o manuales de interés, etc.
2. DESCRIPCION DE FICHEROS DE ENTRADA. Nombre de los ficheros, características físicas (longitud de registros, factor de bloqueo), organización (tipo de registros, forma de archivo, etcétera), diseño del fichero (registros), procedimientos de actualización, etc.
3. DESCRIPCION DEL TRATAMIENTO. Resumen descriptivo, ordinograma, lenguaje a utilizar, ficheros intermedios, utilización de programas estándar, evaluación de ocupación de memoria, tiempos de proceso, etcétera.
4. DESCRIPCION DE LAS SALIDAS. Ficheros de resultados, mantenimiento de estos ficheros, ficheros de seguridad, diseño de salidas impresas (su utilización, sus características, etc.).
5. ANEJOS
   Diseños de registros.
   Procedimientos de cálculo (fórmulas, decimales, redondeos), etc.
   Constantes y tablas a utilizar en los cálculos.
   Codificación: códigos utilizados en el proceso.
   Controles, seguridades.
   Procedimientos de modificaciones futuras, etc.

se utilizan métodos adecuados de representación, concisos y claros (ordinogramas, tablas de decisión, etc.); se diseñan los ficheros, los registros, etc., en formularios preimpresos que ayudan a controlar una información completa y clara; se utilizan tablas de comprobación —checklist (con listas de aspectos a considerar, para no olvidar ningún detalle)—; se desarrollan técnicas específicas de conservación para una eficaz obtención de los datos cuando se estudian (con los usuarios externos) los problemas que se van a mecanizar; se manejan un conjunto de tablas y soluciones típicas para la resolución de numerosas dificultades que suelen surgir, etc.

El conocimiento, aunque sea somero, de estas técnicas entendemos que es sumamente interesante para cualquier persona que desee tener una mínima formación en Informática, pues, aparte de la curiosidad que produce conocer cómo trabajan los profesionales de los grandes Centros de Informática, la mayoría de estos procedimientos son aplicables para el programador individual que utiliza un ordenador personal. En efecto, todas estas técnicas y trucos le facilitarán la definición del problema que Vd. personalmente quiere resolver en su casa o en su oficina, le ayudarán a plasmar en un documento (ordinograma, tabla de decisión, diseño de registro, etc.) la información que ha de manejar (con lo cual simplificará y optimizará su posterior tarea de programación), le ayudarán a ir «documentando» sus trabajos, etc.

Expondremos sucesivamente el tipo de documentos y diseños que se suele manejar en el análisis informático, junto con técnicas y trucos, de preparación de tareas, técnicas de organización de información, listas de comprobación, etc.

TECNICAS DE PROGRAMACION

ALGORITMOS

LOS matemáticos emplean la palabra «algoritmo» para referirse a un conjunto de reglas o pasos bien definidos que permiten realizar un cálculo determinado. Esta palabra proviene del nombre del matemático árabe Abur Jáfar Mohamed Ibn Musa al-Jowarizmi, que vivió en el siglo IX y que fue famoso durante toda la Edad Media.

En realidad los algoritmos no fueron inventados por el científico que les legó su nombre, pues ya se utilizaban en Mesopotamia dos mil años antes de Cristo, donde se disponía de sistemas de numeración relativamente complejos, que permitían realizar operaciones con números fraccionarios y resolver ecuaciones de segundo grado.

Algunos de los algoritmos antiguos se resolvían utilizando maquinarias especiales. Una de las más sencillas y flexibles era el ábaco, inventado probablemente en Oriente Próximo durante el segundo milenio antes de Cristo y perfeccionado en China dos mil años después de su origen. El ábaco permite realizar operaciones aritméticas sencillas (sumas, restas, multiplicaciones y divisiones) mediante reglas bien definidas, que pueden llevarse a la práctica con gran rapidez.

Un algoritmo puede representarse esquemáticamente de muchas maneras diferentes, aunque equivalentes entre sí. Por ejemplo, mediante un conjunto de reglas numeradas como las siguientes:

1. Tomamos un número entero que llamaremos X.
2. Dividimos X entre 2 y calculamos el cociente y el resto.
3. Si el cociente no es cero, sustituimos X por el cociente y volvemos al paso 2.
4. Si el cociente es cero, damos como solución la sucesión de todos los restos obtenidos en orden inverso.

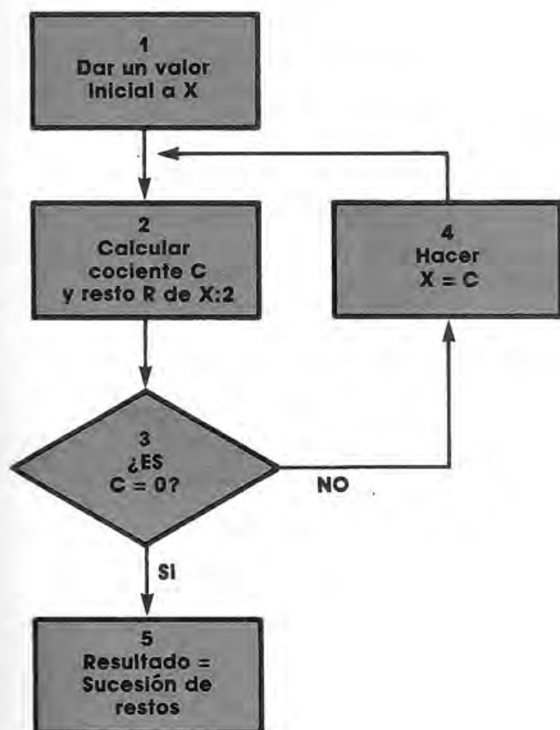
Veamos un ejemplo de aplicación del algoritmo anterior:

1. Sea X igual a 13.
2. Al dividir X entre 2 obtenemos el cociente 6 y el resto 1.
3. Como el cociente no es cero, hacemos X igual a 6 y repetimos el paso 2.
4. Al dividir 6 entre 2 obtenemos el cociente 3 y el resto 0.
5. Como el cociente no es cero, hacemos X igual a 3 y repetimos el paso 2.
6. Al dividir 3 entre 2 obtenemos el cociente 1 y el resto 1.
7. Como el cociente no es cero, hacemos X igual a 1 y repetimos el paso 2.
8. Al dividir 1 entre 2 obtenemos el cociente 0 y el resto 1.
9. Como el cociente es cero, damos como solución la sucesión de los restos obtenidos en orden inverso (1 1 0 1) y terminamos la ejecución del algoritmo.

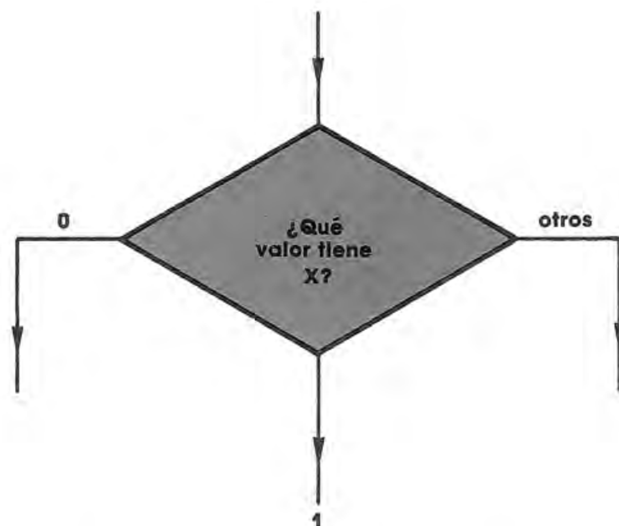
Como se observará, el algoritmo anterior permite convertir un número cual-

quiera al sistema de numeración de base 2. Una modificación trivial del mismo algoritmo permite convertir números decimales a cualquier base.

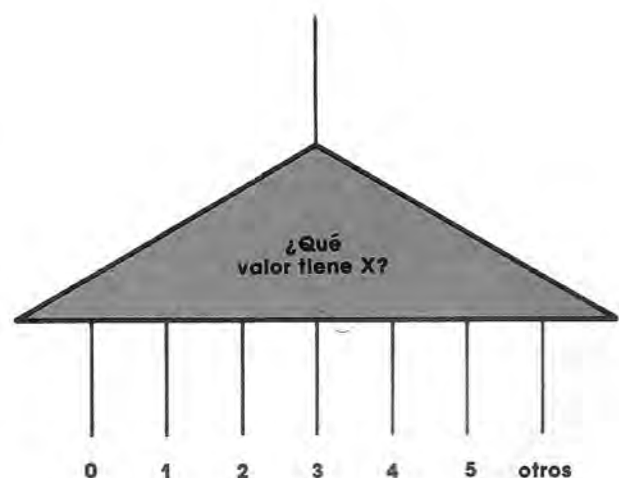
Los algoritmos pueden describirse también mediante organigramas, también llamados diagramas de bloques, que son representaciones pictóricas de las operaciones a realizar. Veamos, por ejemplo, el organigrama equivalente al algoritmo anterior:



En este organigrama, las cajas rectangulares representan acciones a realizar, llegándose a ellas por una entrada única y saliéndose, asimismo, una vez terminada la operación correspondiente, por un salida única. Los rombos, por el contrario, no contienen acciones, sino preguntas cuya contestación representa la decisión que hay que tomar para continuar la marcha del programa. Por tanto, un rombo tiene siempre más de una salida. En el caso más sencillo, el de la figura anterior, la pregunta sólo podrá responderse afirmativa o negativamente, por lo que las salidas serán dos, correspondiendo una de ellas a la respuesta SI y la otra a la respuesta NO. Sin embargo, también es posible hacer preguntas más complicadas o que puedan tener más de dos respuestas diferentes. Veamos algunos ejemplos:

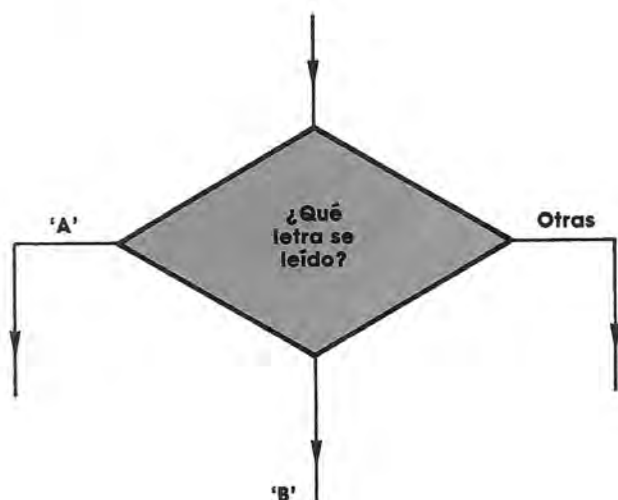


donde se tomará la salida de la izquierda si el valor de X es igual a cero, la del centro si vale 1 y la de la derecha si tiene cualquier otro valor. Como se observará, el punto de bifurcación anterior da lugar a tres posibilidades diferentes, pero es fácil comprender que este tipo de preguntas pueda dar lugar a un número ilimitado de salidas posibles, como en el caso siguiente:



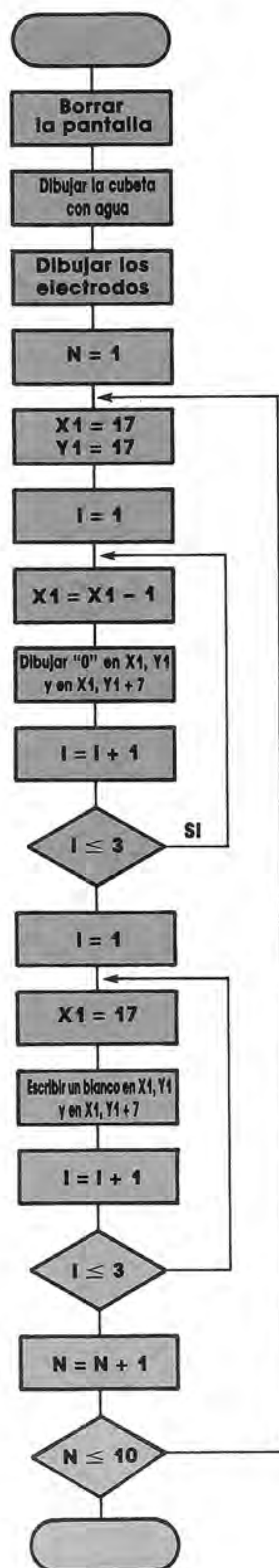
Se observará que en este caso hemos abandonado la forma rómbica para adoptar la triangular, que permite dibujar un número mayor de salidas en el bloque de bifurcación.

Veamos, por último, el caso de una pregunta que puede responderse de dos o más formas posibles, diferentes del simple SI o NO.



Hemos mencionado dos tipos de bloque: los rectangulares, que especifican una acción, y los rómbicos o triangulares, que definen los puntos de bifurcación condicional. Con ellos se pueden construir organigramas o diagramas de bloques para todo tipo de programas, por complicados que éstos sean. En la práctica, pueden utilizarse también otros tipos de bloques diferentes para simplificar el dibujo y hacerlo más legible (bloques de comienzo o fin de programa, bloques de llamada de una subrutina, bloques que permiten enlazar entre sí hojas distintas de un organigrama muy grande), pero ninguno de éstos es estrictamente necesario.

Para terminar, veamos el diagrama de bloques de un par de programas completos: el primero dibuja en la pantalla una cubeta llena de agua en la que están introducidos dos electrodos, por los que se supone circula una corriente eléctrica, pues de ellos surgen burbujas que ascienden a través del agua y desaparecen rápidamente en el exterior. El diagrama de bloques correspondiente es el que se indica a continuación:



Obsérvese que, además de los bloques rectangulares que representan acciones y de los rombos que indican decisiones a tomar, existen en este diagrama dos bloques curvos, situados en sus extremos superior e inferior, respectivamente, y que no tienen otro objeto que indicar claramente cuál es el principio y el final del programa. En general, este organigrama permanecerá invariable, cualquiera que sea el lenguaje de programación que vayamos a utilizar para construir el programa, con una excepción que veremos más adelante. Veamos, por ejemplo, una versión del mismo bloque en lenguaje BASIC:

```
5 REM Dibujo de un tubo lleno de agua con dos electrodos
10 CLS
20 LOCATE 10,10:PRINT CHR$(191)+SPACE$(20)+CHR$(218)
30 LOCATE 11,10:PRINT CHR$(179)+SPACE$(20)+CHR$(179)
40 LOCATE 12,10:PRINT CHR$(179)+SPACE$(20)+CHR$(179)
50 LOCATE 13,10:PRINT CHR$(195)+STRING$(20,196)+CHR$(180)
60 LOCATE 14,10:PRINT CHR$(179)+SPACE$(20)+CHR$(179)
70 LOCATE 15,10:PRINT CHR$(179)+SPACE$(20)+CHR$(179)
80 LOCATE 16,10:PRINT CHR$(179)+SPACE$(20)+CHR$(179)
90 LOCATE 17,10:PRINT CHR$(179)+SPACE$(20)+CHR$(179)
100 LOCATE 18,10:PRINT CHR$(192)+STRING$(20,196)+CHR$(217)
105 REM Dibujo de dos electrodos
110 LOCATE 8,16:PRINT CHR$(218)+STRING$(12,196)+CHR$(180)+" "+CHR$(191)
120 LOCATE 9,16:PRINT CHR$(179)+SPACE$(8)+CHR$(218)+STRING$(5,196)+CHR$(217)
130 LOCATE 10,16:PRINT CHR$(179)+SPACE$(8)+CHR$(179)
140 LOCATE 11,16:PRINT CHR$(179)+SPACE$(8)+CHR$(179)
150 LOCATE 12,16:PRINT CHR$(179)+SPACE$(8)+CHR$(179)
160 LOCATE 13,16:PRINT CHR$(197)+STRING$(8,196)+CHR$(197)
170 LOCATE 14,16:PRINT CHR$(179)+SPACE$(8)+CHR$(179)
180 LOCATE 15,16:PRINT CHR$(179)+SPACE$(8)+CHR$(179)
190 LOCATE 16,16:PRINT CHR$(179)+SPACE$(8)+CHR$(179)
200 REM Burbujitas
205 FOR N=1 TO 10
210 X1=17:Y1=17
220 FOR I=1 TO 3:X1=X1-1:LOCATE X1,Y1: PRINT "o":LOCATE X1,Y1+7:PRINT"o":NEXT
230 X1=17
240 FOR I=1 TO 3:X1=X1-1:LOCATE X1,Y1: PRINT " ":LOCATE X1,Y1+7:PRINT" ":NEXT
250 NEXT N
```

Comparando el programa con el organigrama se verá que un bucle FOR-NEXT puede representarse también haciendo uso únicamente de los bloques rectangulares y de los rombos. Volveremos sobre esto más adelante.

Al contrario del organigrama, este programa depende no ya sólo del lenguaje de programación en que ha sido escrito (BASIC en este caso), sino también de la máquina en que vaya a ejecutarse. La versión anterior corresponde, en concreto, al ordenador personal de IBM y compatibles, pero no producirá los resultados apetecidos en otras máquinas diferentes. Esto se debe a que las expresiones del tipo CHR$(218) representan un carácter gráfico en el IBM PC, pero pueden corresponder a otro carácter totalmente diferente en el SPECTRUM, a un tercero en

AMSTRAD, etc. Veamos cómo se modificaría este mismo programa para otros ordenadores:

Por último, vamos a ver el organigrama de un programa que calcula la tabla de interés compuesto que puede obtenerse

```
  AMSTRAD: Cambiar todos los
CHR$(191) por CHR$(156)
CHR$(218) por CHR$(150)
CHR$(179) por CHR$(149)
CHR$(195) por CHR$(151)
CHR$(180) por CHR$(157)
CHR$(192) por CHR$(147)
CHR$(217) por CHR$(153)
CHR$(197) por CHR$(159)
```

```
  COMMODORE: Cambiar todos los
CHR$(191) por CHR$(174)
CHR$(218) por CHR$(176)
CHR$(179) por CHR$(98)
CHR$(195) por CHR$(171)
CHR$(180) por CHR$(179)
CHR$(192) por CHR$(173)
CHR$(217) por CHR$(189)
CHR$(197) por CHR$(123)

  En todos los sitios donde pone
SPACE$(20)
  se pondrá
FOR Z=1 TO 20: PRINT " ";: NEXT Z

  En todos los sitios donde pone
SPACE$(8)
  se pondrá
FOR Z=1 TO 8: PRINT " ";: NEXT Z

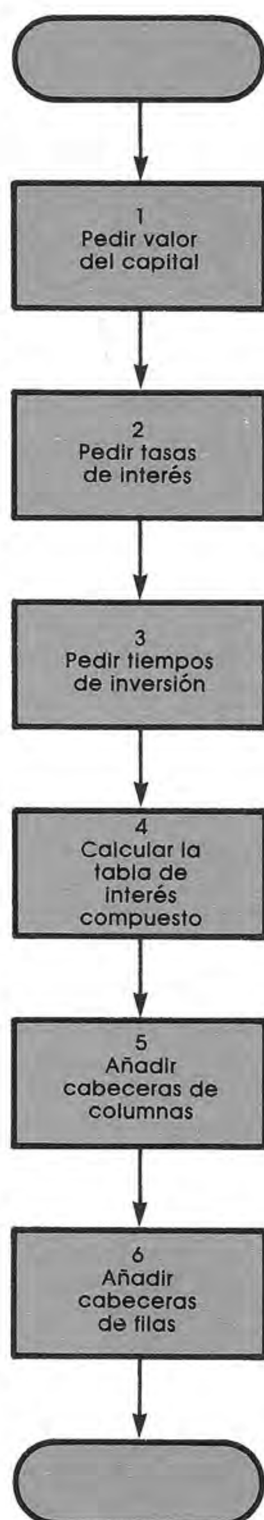
  En todos los sitios donde pone
STRING$(20,196)
  se pondrá
FOR Z=1 TO 20: PRINT CHR$(99);: NEXT Z

  En todos los sitios donde pone
STRING$(12,196)
  se pondrá
FOR Z=1 TO 12: PRINT CHR$(99);: NEXT Z

  También hay que cambiar todas las sentencias LOCATE.
  Así, si vemos que en el listado pone
LOCATE 17,10
  hay que sustituirlo por
POKE 214,17: POKE 211,10
```

```
  MSX: Cambiar el orden en todas las sentencias LOCATE.
  Así, si aparece en el listado
LOCATE 12,10
  hay que sustituirlo por
LOCATE 10,12
```

partiendo de un capital determinado, y aplicándole distintos tipos de interés durante diversos períodos de tiempo.



Se observará que este diagrama es totalmente lineal, pues se compone de un conjunto de acciones consecutivas que se suceden unas a otras sin bifurcación alguna. Es evidente, sin embargo, que alguno de los bloques (como el que calcula la tabla de interés compuesto) corresponde a la realización de una acción relativamente compleja, que bajo ciertas condiciones será necesario especificar con más cuidado, descomponiéndola en un organigrama más detallado. Cuando hablemos de programación modular volveremos sobre este punto. Sin embargo, esto no será necesario en este caso concreto, pues el lenguaje que vamos a utilizar (APL) para construir el programa correspondiente al organigrama de la figura es lo bastante potente como para que el cálculo de la tabla completa de interés compuesto pueda realizarse en una sola instrucción, por lo que no será necesario aumentar el detalle del diagrama de bloques

Esta es, precisamente, la excepción a la regla de que hablábamos un poco más arriba. Es verdad que, en general, el organigrama puede ser independiente del lenguaje de programación. Pero cuanto más potente sea éste, menor será el grado de detalle al que habrá de llegar el diagrama para resolver un problema concreto.

El programa representa la realización del diagrama de bloques anterior en el lenguaje APL, junto con el resultado de una ejecución de dicho programa. La correspondencia entre las líneas del programa y los bloques de la figura es como sigue:

Bloque n.º.	Líneas n.º
1	1-2
2	3-4
3	5-6
4	7
5	8-9
6	10-11

```
[0]  RES←INTERES
[1]   'DEME EL VALOR DEL CAPITAL'
[2]   CAPITAL←⎕
[3]   'DEME LOS VALORES DE LAS TASAS DE INTERES'
[4]   TASA←⎕
[5]   'DEME LOS TIEMPOS DE INVERSION'
[6]   TIEMPO←⎕
[7]   VALOR ← CAPITAL × (1+TASA÷100) ∘.* TIEMPO
[8]   RES ← '-' ,[1] 7 0 ⍕ VALOR
[9]   RES ← (7 0 ⍕ TIEMPO),[1] RES
[10]  RES ← '|' , RES
[11]  RES ← (' ',[1]' ',[1] 4 1⍕((⍴TASA),1)⍴TASA),RES


      INTERES
DEME EL VALOR DEL CAPITAL
⎕:
      100000
DEME LOS VALORES DE LAS TASAS DE INTERES
⎕:
      10 10.5 11 11.5 12
DEME LOS TIEMPOS DE INVERSION
⎕:
      4 5 6 7 8
    |      4       5       6       7       8
    |-----------------------------------------
10.0| 146410 161051 177156 194872 214359
10.5| 149090 164745 182043 201157 222279
11.0| 151807 168506 187041 207616 230454
11.5| 154561 172335 192154 214252 238891
12.0| 157352 176234 197382 221068 247596
```

LOGO

INTRODUCCION

TODOS sabemos que, en la actualidad, existen muchos lenguajes de programación, es decir, diferentes formas de decirle al ordenador lo que deseamos que haga. Lo que ocurre es que cada uno de ellos tiene unas características que le hacen ser más utilizado en unas áreas que en otras. En concreto, el LOGO, desde su nacimiento, ha tenido como objetivo el estar orientado a niños y a jóvenes. Por ello, sus principales aplicaciones tienden a ser aquellas relacionadas con este tipo de usuarios.

Así, por ejemplo, es el lenguaje que más se utiliza a la hora de introducir la informática en la escuela. Y esto no es por casualidad o por capricho, sino porque tiene ventajas que otros lenguajes no poseen. Además de ser un lenguaje que resulta bastante fácil de aprender, ayuda a entrar en el mundo de la Informática (a conocer sus conceptos fundamentales, como son los procedimientos, variables, instrucciones, memoria...) de una manera sencilla y casi como si fuera un juego. Por otro lado, no sólo nos permite realizar dibujos, sino que también se puede utilizar para resolver problemas más complicados (algunos parecidos a los que se plantean en el colegio).

Para comprobar que esto es cierto, lo mejor es aprender cómo podemos hacer que el ordenador haga lo que nosotros queramos empleando este lenguaje. Vamos a empezar.



Cómo nos comunicamos con el ordenador

Una vez que nuestro ordenador tiene cargado en su memoria el LOGO (es decir, entiende este lenguaje), podemos darle una serie de órdenes para que haga lo que nosotros queramos. Pero estas órdenes no se las podemos dar de cualquier manera, de la misma forma que a un chino no le podemos hablar en castellano porque no nos entendería. Para que el ordenador comprenda lo que le decimos hay que usar unas palabras especiales llamadas **comandos.**

Cuando queramos que el ordenador se entere de lo que nosotros deseamos que haga se lo tenemos que escribir por medio del **teclado.** El teclado es una especie de máquina de escribir con una serie de teclas que contienen las letras, los números y otros caracteres especiales que luego vamos a necesitar para escribir determinadas órdenes.



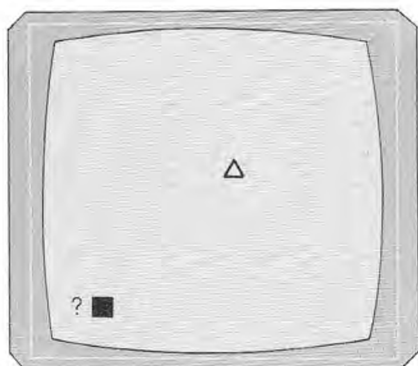
Además, una vez que hemos dado una orden, el ordenador nos ha de enseñar si realmente la ha cumplido o no. Para ello, utiliza la **pantalla,** que es como una televisión en la que no se ven películas o dibujos animados, sino los dibujos o resultados de lo que le hemos mandado hacer.





El mundo de la Tortuga

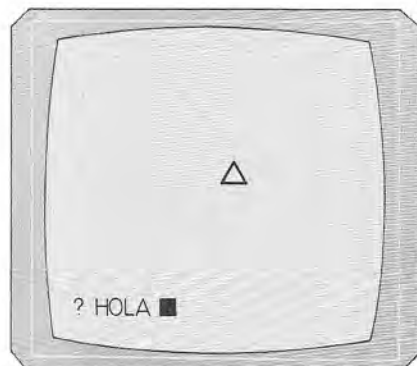
Cuando el ordenador está dispuesto para cumplir las órdenes que le demos usando los comandos del lenguaje LOGO (palabras especiales que él entiende), en la pantalla nos aparecerá lo siguiente:



En el centro tenemos un triángulo (▽) que representa a una **tortuga.** Esta tortuga va a ser la que va a realizar lo que nosotros le digamos. Como podemos ver, su posición inicial es estar en el centro de la pantalla y mirando hacia arriba.

La interrogación (?) nos dice que la tortuga está esperando a que le demos una orden.

Por último, el cuadradito (■) se llama **cursor** y sirve para irnos mostrando lo que nosotros escribamos en el teclado. Por ejemplo, si escribimos **hola** nos quedará:



Sólo nos queda saber cómo decirle a la tortuga que cumpla la orden que le hemos dado. Para ello, existe una tecla que hemos de pulsar cada vez que queramos que la tortuga ejecute un comando. Esta tecla se llama **Return** o **Enter** y en nuestro teclado puede aparecer así:



Ahora que ya sabemos cómo comunicarnos con la tortuga, vamos a empezar a aprender su lenguaje: el LOGO.



Primeros comandos

Nuestra tortuga lleva en su espalda un lápiz que le permite ir pintando cuando anda, es decir, al moverse va dejando un rastro. Aprovechando esto, podemos darle órdenes para que haga dibujos.

Lo primero que le podemos decir es que vaya hacia delante. Para eso, se utiliza el comando

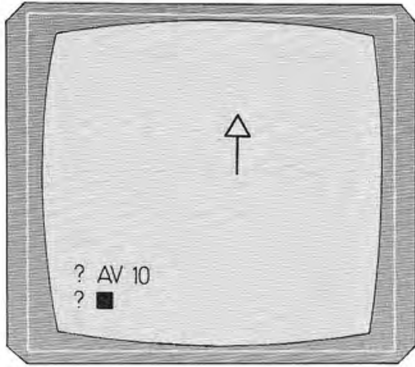
avanza n

o en abreviatura

av n

donde **n** es el número de pasos que queremos que dé.

Así, si le decimos **av 10** y damos a la tecla de **Return,** en la pantalla aparecerá:



La tortuga ha ejecutado nuestra orden y espera que le demos otra.

Si ahora queremos que vaya hacia la derecha, le hemos de decir que tuerza hacia este lado. Para ello, el comando que hemos de usar es:

giraderecha m

o en abreviatura

gd m

siendo **m** el número de grados que queremos que gire.

Por el contrario, si queremos que vaya hacia la izquierda, el comando es:

giraizquierda m

o

gi m

Por último, si en lugar de avanzar queremos que ande hacia atrás, el comando es:

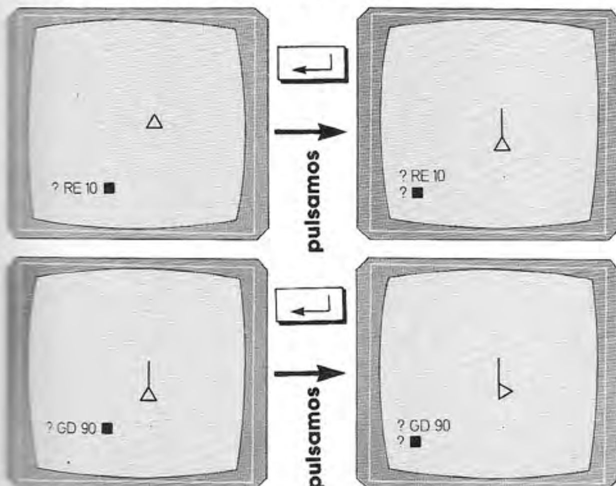
retrocede n

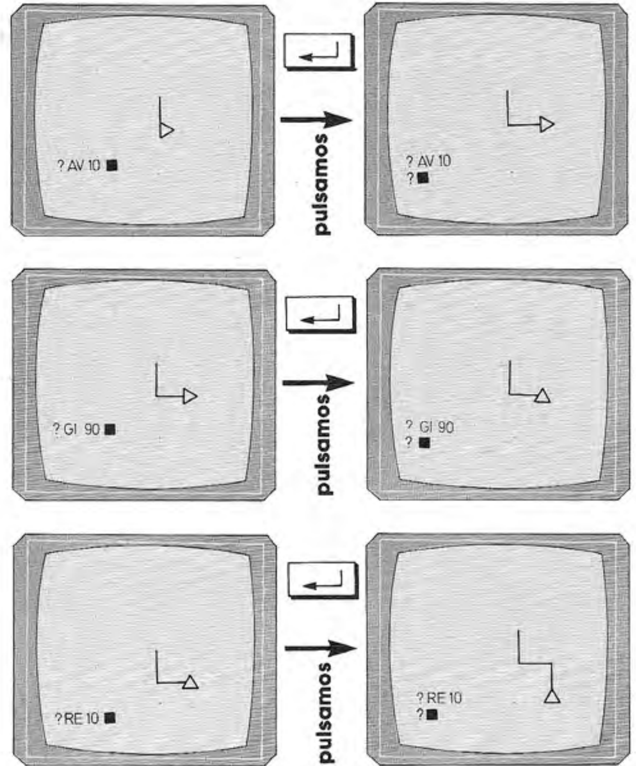
o

re n

siendo **n** el número de pasos.

Veamos el resultado de ejecutar algunos comandos de este tipo:

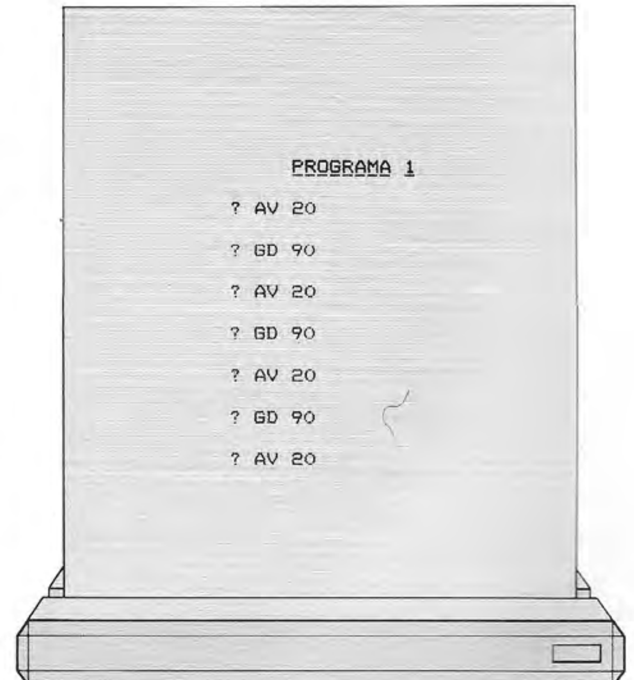




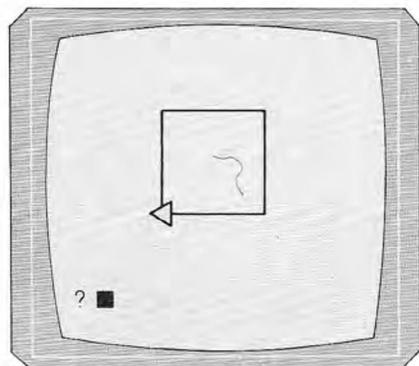


Primeros dibujos

Vamos a decirle a la tortuga que pinte un cuadrado. Para ello, le tenemos que dar las siguientes órdenes:



Como resultado tendremos:



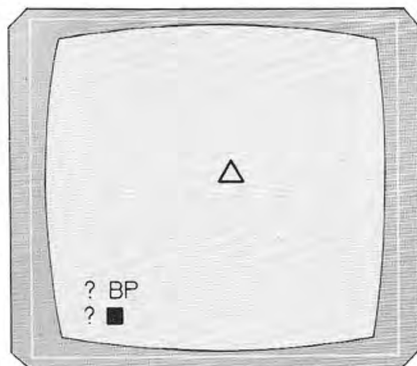
Si ahora queremos dibujar un rectángulo, primero hemos de borrar el dibujo anterior. Para decirle a la tortuga que borre todo lo que haya pintado en la pantalla y vuelva a su posición original (en el centro y mirando hacia arriba) usamos el comando

borrapantalla

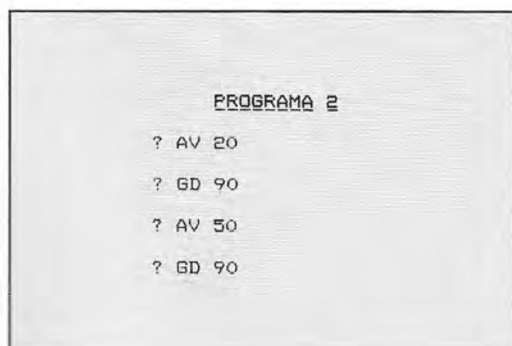
o

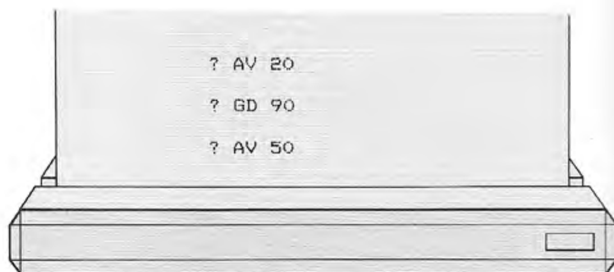
bp

Tras mandar a la tortuga que lo ejecute nos quedará:

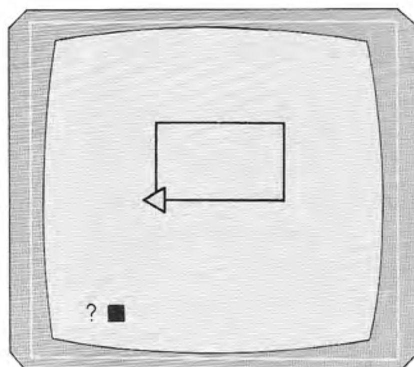


Ya podemos dibujar nuestro rectángulo. Para ello, escribimos lo siguiente:





y como resultado tenemos:



Si queremos que la tortuga vuelva a su posición inicial sin borrar nuestro dibujo, hemos de escribir el comando:

centro

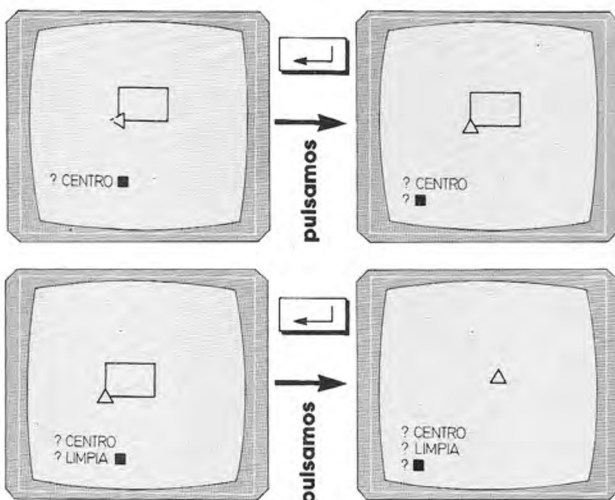
Por el contrario, si queremos que la tortuga borre el dibujo, pero que no vuelva a su posición inicial, el comando es:

limpia

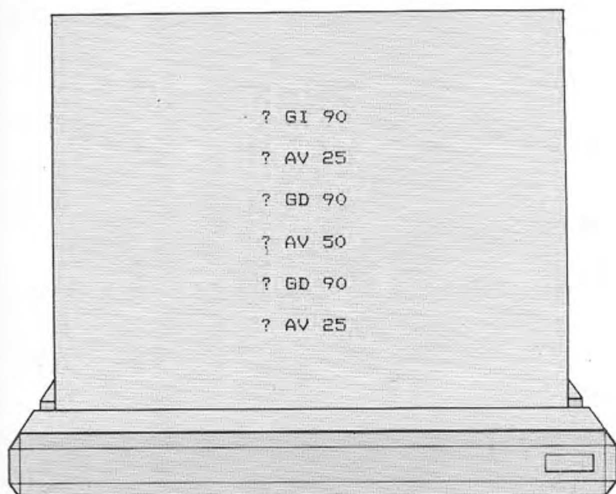
Por tanto, si damos la orden **bp,** esto es equivalente a decir primero **centro** y luego **limpia:**

BP = CENTRO + LIMPIA

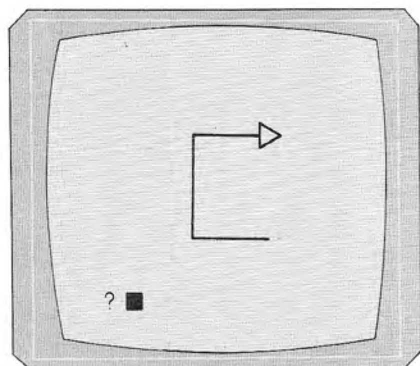
Para comprobarlo:



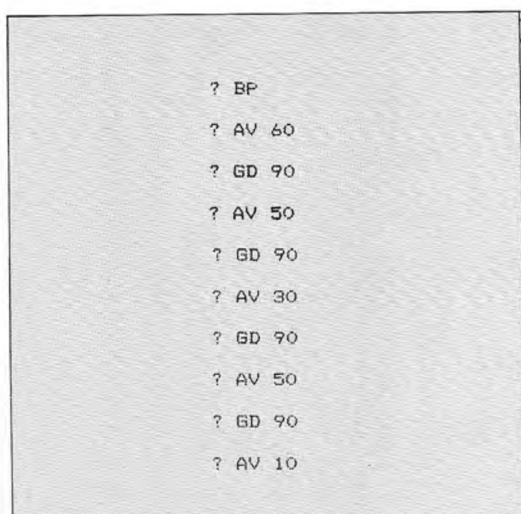
Vamos a hacer que ahora la tortuga pinte una letra, por ejemplo, la "C". Para ello, le damos las siguientes órdenes:

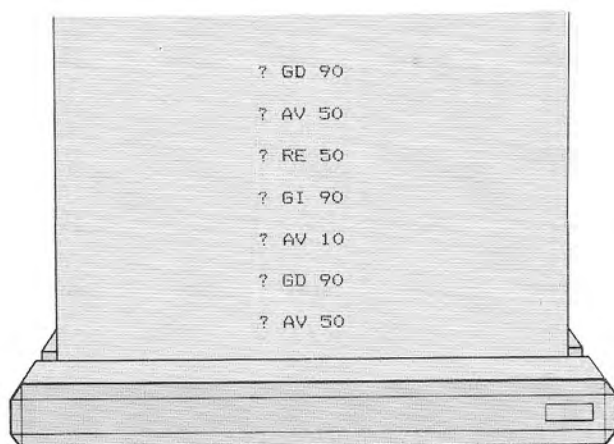


Ya está. Nos ha quedado:

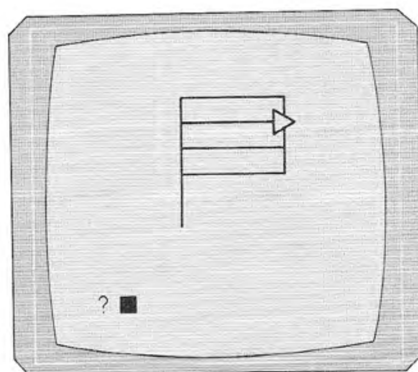


Ahora vamos a dibujar una bandera. Hemos de escribir lo siguiente:





y nos queda:

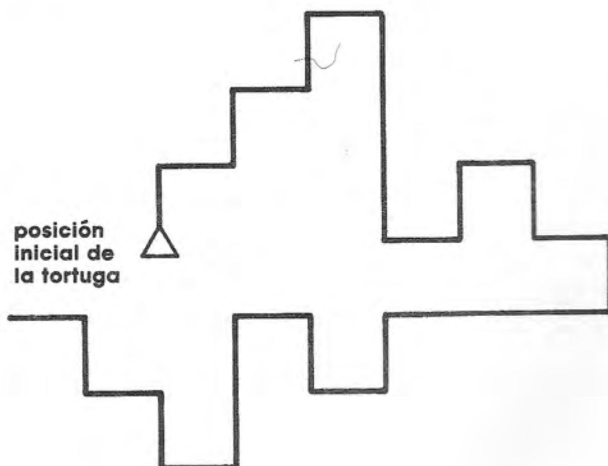


Ahora os toca a vosotros. Podéis dibujar lo que se os ocurra.

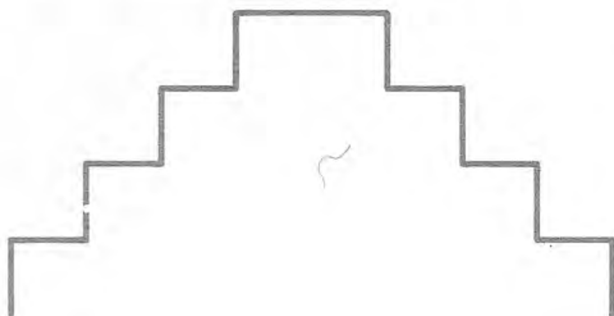


Os proponemos

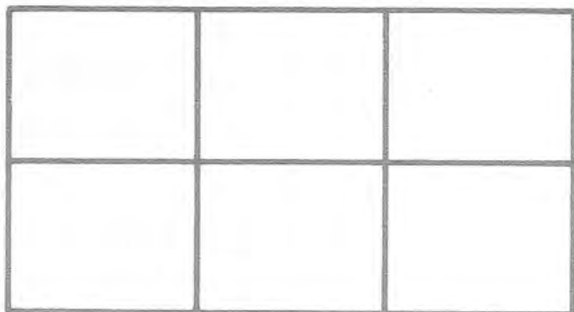
1. Haz que la tortuga pinte este laberinto:



2. Ahora pinta una escalera:



3. Intenta una ventana:



4. Pinta un cuadrado usando sólo los comandos GI y RE.

5. También puedes hacer que dibuje los números:



6. Ahora algunas letras:





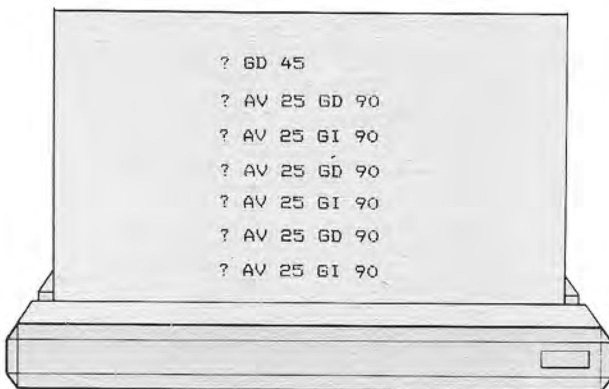
Giros más complicados

Hasta ahora, en todos los dibujos que hemos realizado los giros de la tortuga eran de 90 grados. Pero existen figuras en las que hemos de hacer líneas oblicuas. Para ello, la tortuga debe avanzar o retroceder de una forma inclinada.

Vamos a verlo con un ejemplo. Supongamos que queremos dibujar unas montañas de esta forma:



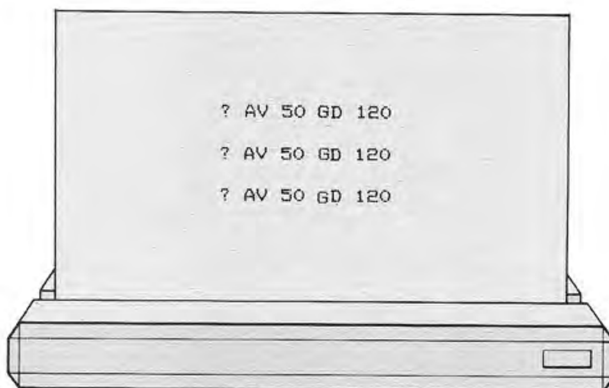
Como vemos, para empezar, la tortuga no ha de girar 90, sino menos. Por tanto, el conjunto de comandos serían:



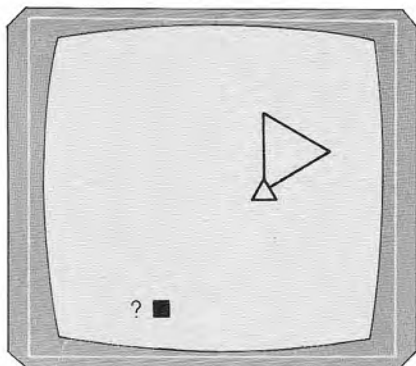
Como se puede ver, podemos escribir en una misma línea varios comandos, siempre y cuando dejemos al menos un espacio en blanco entre ellos.

Podemos pintar ahora un triángulo. Los ángulos de un triángulo son de 60 grados. Pero si hacemos que la tortuga gire este número de grados no nos saldrá esta figura. Hemos de girar 120. Más adelante veremos por qué.

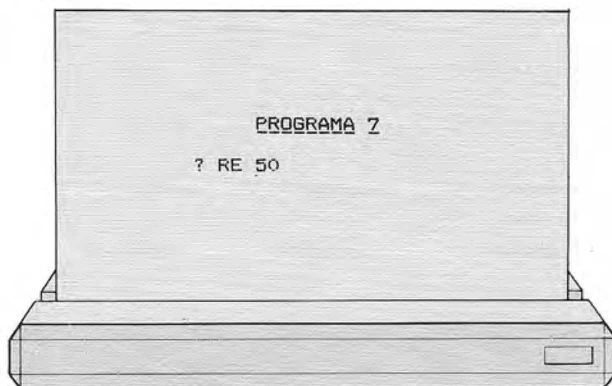
Por tanto, tenemos que dando estas órdenes:



nos sale la figura que queríamos:



Si ahora hacemos



Cómo cargar el Logo

El Logo, al igual que el Basic, es un lenguaje de programación. Por ello es necesario que el Logo se encuentre en la memoria del ordenador. Si no lo estuviese el ordenador no sabría Logo y no podría entendernos.
Para cargar el Logo sigue las instrucciones:

SPECTRUM:

1. Teclea en tu ordenador LOAD "" y pulsa la tecla ENTER.
2. Introduce la cinta de Logo en tu lector de cassette y pulsa la tecla PLAY.
3. El Logo se cargará y ejecutará automáticamente.

MSX:

Si tienes el cartucho:
1. Apaga tu ordenador.
2. Conecta el cartucho de Logo en el SLOT de expanxión.
3. Conecta el ordenador.
4. El Logo aparece automáticamente.

Si tienes la cinta:

1. Teclea en tu ordenador CLOAD "",R y pulsa RETURN.
2. Introduce la cinta de Logo en el cassette y pulsa la tecla PLAY.
3. El Logo se cargará y ejecutará automáticamente.

IBM:

Para cargar el Logo en el IBM tienes que meter primero el disquete del MS/DOS. Una vez que lo hayas hecho, lee las instrucciones que acompañan al disquete de Logo para poder cargarlo.

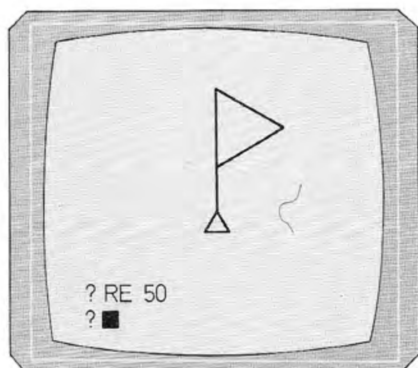
AMSTRAD:

Introduce el disquete con el sistema operativo CP/M. Cuando éste se haya cargado teclea la palabra Logo y éste se cargará y ejecutará automáticamente.

COMMODORE:

El Commodore no tiene Logo en español. El único Logo que hay para Commodore está en inglés.
Aunque con estas instrucciones puedes cargar el Logo, siempre es conveniente que leas las instrucciones que te dieron con el programa para no cometer errores.

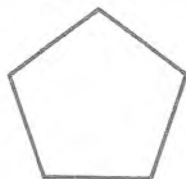
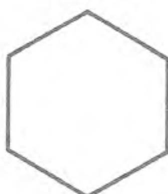
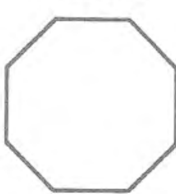
nos queda un banderín:



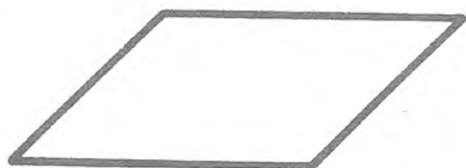
Ahora puedes dibujar muchas más cosas que antes.

Os proponemos

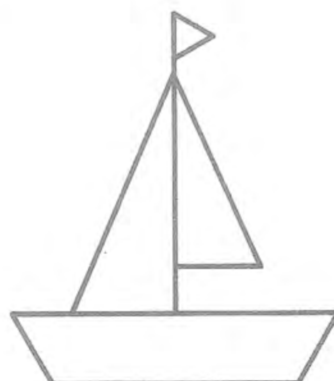
1. Intenta dibujar estas figuras geométricas. Como pista te damos el ángulo que la tortuga ha de girar:

pentágono	hexágono	octógono
		
72	60	45

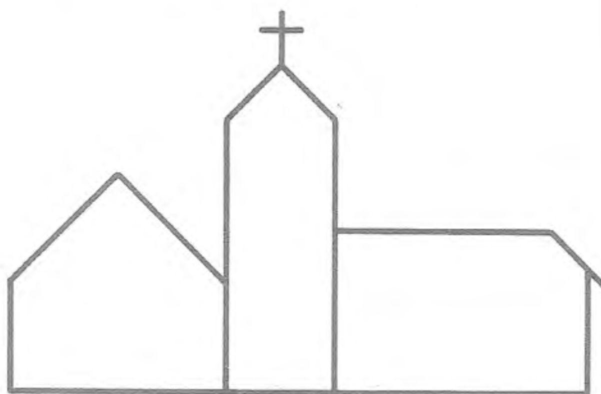
2. Ahora sin pistas, pinta un paralelogramo:



3. Dibuja un barco:



4. Puedes dibujar un pueblo:



5. También puedes pintar letras:

KMNQRVWXYZ

PASCAL

EL PASCAL es el lenguaje más popular de los denominados «estructurados». Fue creado por el suizo Niklaus Wirth allá por el año 1970.

A los programadores acostumbrados a otros lenguajes no estructurados como el BASIC o el FORTRAN, lo que les suele llamar la atención a primera vista del PASCAL es la nula, o casi nula, utilización de instrucciones del tipo GOTO, aunque, como se irá viendo a lo largo de la colección, esto no es más que la punta del iceberg.

También suele dar la sensación de que para hacer un programa en PASCAL hay que escribir mucho «rollo»; sin embargo, veremos que ello, lejos de ser un inconveniente, contribuye a hacer los programas mucho más claros y a disminuir el número de errores que se cometen al prepararlos.

Precisamente es éste el principal objetivo de los lenguajes estructurados, entre los que se encuentran, por ejemplo, además del PASCAL, el ADA y el MODULA-2.

El motivo de que, hasta fechas recientes, el BASIC haya sido el lenguaje más difundido entre los ordenadores personales se debe a que, dada su simpleza —lo que supone una menor necesidad de memoria— y el hecho de que es un lenguaje «interpretado», los fabricantes de ordenadores personales lo escogieron para incorporarlo de fábrica a sus máquinas. Así, para programar en BASIC, normalmente lo único que hay que hacer es simplemente encender el ordenador y empezar a teclear el programa.

Por el contrario, hasta hace pocos años, para programar en PASCAL, como normalmente es un lenguaje del tipo «compilado», en general había que pasar por varias etapas.

En primer lugar había que teclear el programa; para ello se utilizaba un «editor», que es el tipo de programa que se emplea para escribir, por ejemplo, cartas con ayuda del ordenador o este mismo artículo. El texto del programa así preparado se guardaba entonces en un «fichero» en disco o cinta.

A continuación, se procesaba ese fichero para traducirlo al lenguaje que realmente entiende el ordenador, que es el denominado «código máquina». Para ello se utilizaba un programa «compilador» o traductor de PASCAL.

Por último (y a veces tras un proceso adicional llamado en jerga informática «link-edición»), la traducción así obtenida era entregada al ordenador para su ejecución.

Esta pesada manera de proceder, que es la habitual con la mayoría de los lenguajes en los ordenadores grandes, tampoco ayudó mucho a la difusión del PASCAL.

Hoy día, sin embargo, existen para la mayoría de ordenadores personales programas editores-compiladores de PAS-

CAL que, una vez cargados en la memoria del ordenador, permiten desarrollar los programas de una manera prácticamente tan cómoda y rápida como con el BASIC, pero conservándose toda la potencia y enormes ventajas del PASCAL tradicional.

En concreto, hay que citar el TURBO-PASCAL de la casa Borland que, funcionando en los ordenadores personales IBM y compatibles, se ha convertido en el programa compilador más vendido de la historia de la informática y ha contribuido decisivamente a convertir al PASCAL en el lenguaje más popular de la actualidad.

Así pues, lo primero que tiene que hacer el lector es proveerse del programa compilador más adecuado para su ordenador. Todos ellos suelen traer en su manual claros ejemplos ilustrando cómo proceder desde la escritura del programa hasta que ese programa es puesto en marcha. Dada la gran variedad de compiladores existentes, nos procuraremos ceñir a lo largo de la colección a una versión de PASCAL más o menos estándar, para que así los programas de ejemplo funcionen en todos los casos con el mínimo de cambios posible, sin perjuicio de que alguna vez se hagan incursiones en las peculiaridades de algún ordenador en concreto.



Nuestro primer programa en PASCAL

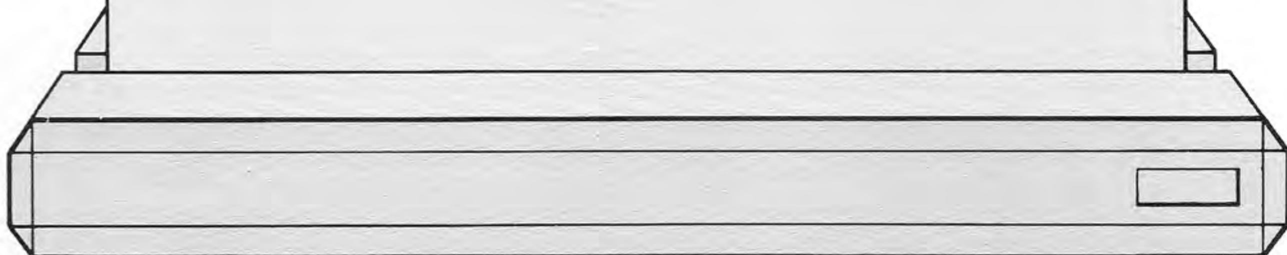
Veamos un programa muy corto:

```
program Contador;


 (* Este programa presenta números desde 1 hasta el valor de TOPE *)
 (* al mismo tiempo, nos enseña los cuadrados de esos números     *)


 const
   Tope = 10;
 var
   Numero : integer;


BEGIN
   writeln ('El valor limite es ',Tope);
   writeln;
   for Numero:=1 to Tope do
     writeln (Numero:3,' al cuadrado es ',Numero * Numero);
   writeln;
   writeln ('FIN DE LA CUENTA.')
END.
```



Tras escribir el programa y seguir los pasos exigidos por nuestro compilador para ponerlo en marcha, saldrá lo siguiente en la pantalla del ordenador:

```
El valor límite es 10

  1 al cuadrado es 1
  2 al cuadrado es 4
  3 al cuadrado es 9
  4 al cuadrado es 16
  5 al cuadrado es 25
  6 al cuadrado es 36
  7 al cuadrado es 49
  8 al cuadrado es 64
  9 al cuadrado es 81
 10 al cuadrado es 100

FIN DE LA CUENTA.
```

Palabras reservadas

En el programa hay unas palabras que está claro que no las hemos escogido nosotros: program, const, var, integer, begin... Son lo que se denomina «palabras reservadas», es decir, palabras con un significado especial que no las podemos utilizar para lo que se nos antoje.

En general, estas palabras las escribiremos con minúsculas (aunque eso va al gusto de cada uno), pero ¡ojo!, puede resultar que nuestro compilador sea muy intolerante y nos exija escribirlas todo con mayúsculas, o todo con minúsculas; para salir de dudas lo mejor es mirar un ejemplo cualquiera en el manual.

Identificadores

También hay otras palabras escogidas por nosotros para dar nombre a algunas cosas del programa: Contador, Tope y Numero; las palabras de este tipo se llaman «identificadores».

Los identificadores se pueden escoger al gusto de cada uno, pero han de empezar por una letra, tras la cual se puede poner cualquier combinación de letras y cifras sin espacios en blanco por medio.

V, BITIO, K2R, SYJ38

serían identificadores válidos pero

3Z, P/Q, AVI ON

serían incorrectos y al intentar compilar un programa con ellos el ordenador nos avisaría de la existencia de un error.

Normalmente procuraremos escribir los identificadores con minúsculas excepto la primera letra, pero hay que hacer la misma advertencia de antes: a la mayoría de los compiladores les da lo mismo que se pongan mayúsculas o minúsculas, pero primero hay que cerciorarse. Por otra parte, aunque en principio se pueden escoger palabras larguísimas, cada compilador se suele fijar sólo en los ocho o diez primeros caracteres; por ello, pudiera resultar que para él las palabras Esternon y Esternones fueran una misma cosa.

El lector atento se habrá hecho ya la siguiente pregunta: ¿y qué pasa con eso de «Este programa...» o «El valor...»?

Es evidente que no son palabras reservadas ni identificadores; como veremos en otro momento, son comentarios y textos y, en principio, podemos escribirlos como más nos apetezca, sin restricción alguna.

Para terminar, la palabra «writeln» es un caso especial; es del tipo llamado «identificador predefinido», es decir, tiene un significado especial (sirve para escribir cosas en la pantalla), PERO la podríamos utilizar para algo distinto (aunque, lógicamente, ya no serviría para lo de antes). Sin embargo, y para no complicarnos la vida, los identificadores predefinidos los vamos a considerar como palabras reservadas y no volveremos a hablar de ellos (al menos hasta que no sepamos mucho más PASCAL).



El programa más corto

El programa más simple imaginable en PASCAL sería el siguiente:

```
program MasCorto;
begin
end.
```

y si lo intentamos compilar no habrá ningún problema, aunque al ejecutarlo descubriremos sin sorpresa que no sirve absolutamente para nada.

Los programas, en general, sirven para procesar datos según una serie de ins-

trucciones. Un programa PASCAL tiene por ello dos zonas bien diferenciadas para describir los datos y las instrucciones, quedando siempre su estructura así:

```
program NombreDelPrograma;
```

Zona de descripción de datos

```
begin
```

Zona de instrucciones

```
end.
```

Es decir, siempre hay una «cabecera» formada por la palabra reservada «program» seguida de algún identificador que dé nombre al programa; la cabecera se termina con un punto y coma.

Tras ella vendría la descripción de los datos; como el programa MasCorto no tiene datos, carece de esta zona.

Por último, vendrían las instrucciones del programa enmarcadas por las palabras reservadas «begin» y «end» que en inglés significan, respectivamente, empezar y terminar. Como el programa MasCorto no tiene instrucciones, end se encuentra justo a continuación de begin. Al igual que un párrafo, un programa acaba siempre con un punto final.

Hay un detalle muy importante del PASCAL: podemos utilizar las líneas que queramos para escribir un programa siempre que, eso sí, no partamos entre dos líneas ninguna palabra; incluso se podría escribir un programa en una sola línea lo bastante larga. También se pueden separar las palabras todo lo que se quiera.

Por ello, probemos a compilar los siguientes programas:

```
program
MasCorto
;
begin
end.
```

```
program MasCorto; begin end.
```

Como ejercicio, podríamos probar a reagrupar las diferentes partes del programa Contador a nuestro gusto buscando siempre, por supuesto, la máxima claridad, y de paso, cambiar los identificadores por otros distintos.

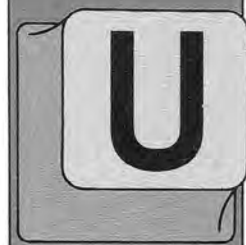
NOTA: Con algunos compiladores la cabecera puede que tenga un aspecto ligeramente distinto, como, por ejemplo:

```
program NombreDelPrograma
(input, output);
```

Si es ese nuestro caso, la pondremos así sin preocuparnos más de ello por el momento.

OTROS LENGUAJES

¿QUE ES UN SISTEMA OPERATIVO?

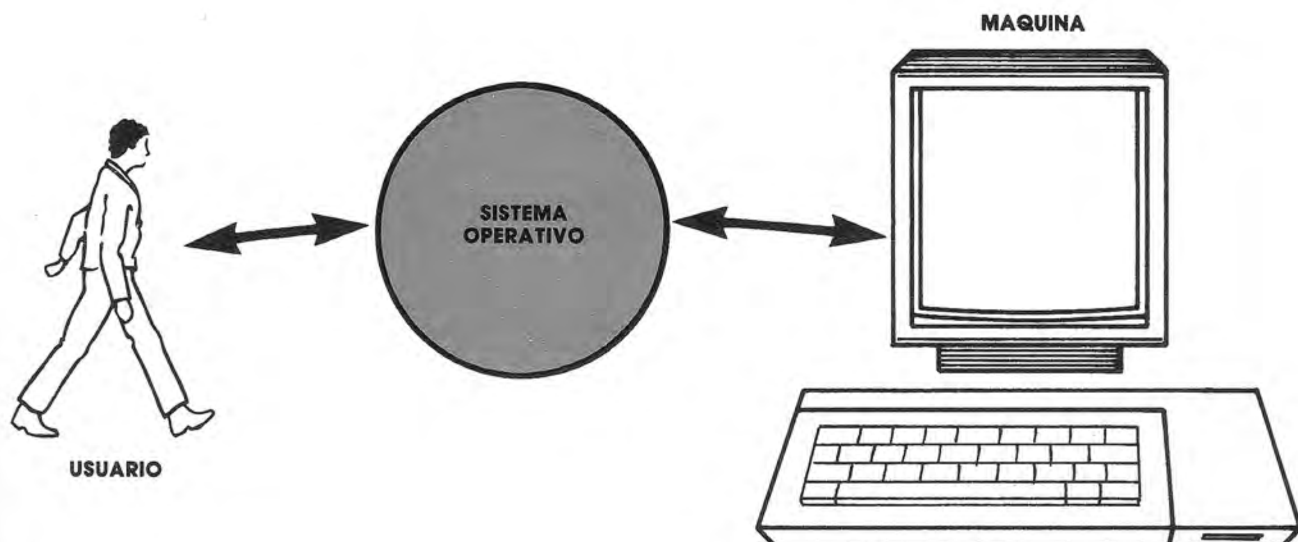


UN sistema operativo es un conjunto de programas que facilitan la comunicación entre la parte puramente física del ordenador (el hardware) y el usuario, a la vez que explotan al máximo los recursos y posibilidades de la máquina para que su uso sea el óptimo.

Básicamente, los cometidos de un sistema operativo se pueden englobar en tres grupos:

— **Gestión del propio sistema ordenador,** es decir, supervisar y controlar tanto el funcionamiento de la Unidad Central de Proceso como de los dispositivos periféricos (pantalla, impresora, disquette, disco duro, etc.).

— **Gestión de las tareas que se le encargan a la máquina,** lo cual incluye:





El usuario no se comunica directamente con la máquina, sino que lo hace con el sistema operativo, y éste con el ordenador.

• **Planificación de los trabajos,** para conseguir una utilización del sistema lo más eficiente posible. Para ello se asignan prioridades a los trabajos encomendados, de manera que se ejecutan primero los de más alta prioridad, normalmente con poca utilización de los recursos del sistema (tiempo de procesador, espacio de memoria y dispositivos periféricos), y se dejan los últimos los trabajos de baja prioridad y que necesitan de una utilización más exhaustiva de esos recursos.

• **Establecer y supervisar las comunicaciones con el entorno,** tanto si son de entrada (carga de programas y datos) como si son de salida (entrega de resultados).

• **Asignación de los recursos del sistema** a los distintos programas en ejecución, que compiten entre ellos por tales recursos y podrían provocar por tal motivo situaciones de conflicto, que deben ser solucionadas por un "árbitro".

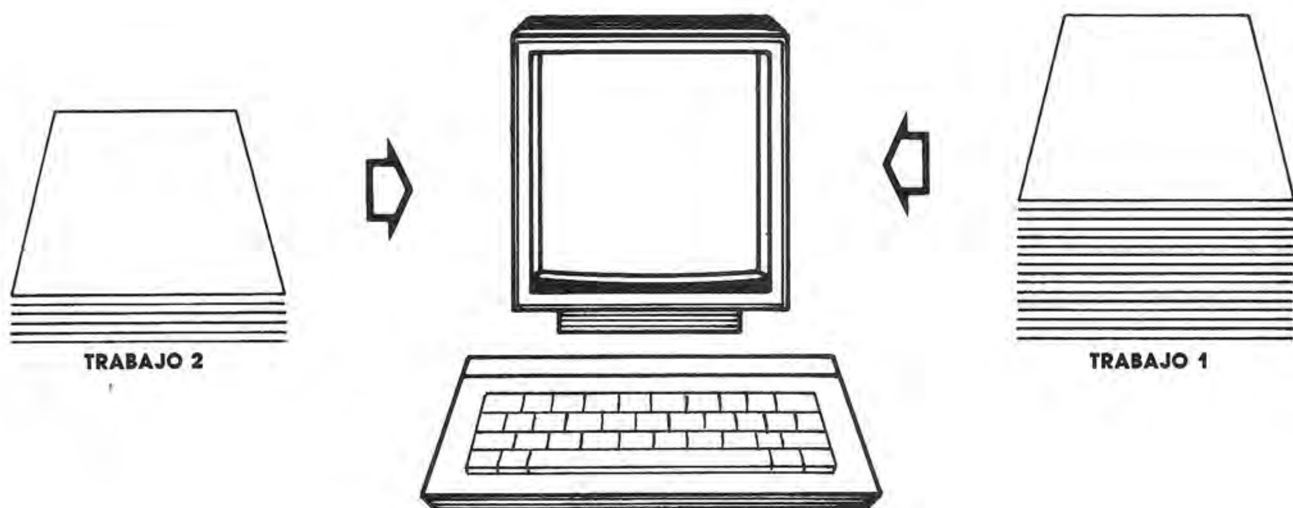
— **Gestión de los datos,** que incluye controlar el sistema de ficheros tanto en su estructura como en el acceso a los mismos, así como los movimientos de los datos entre los dispositivos de E/S y la memoria principal.



Distintos tipos de sistemas operativos

Según los modos de operación de un sistema operativo podemos clasificarlos en varios tipos esenciales, aunque algunos sistemas actuales permiten a un ordenador trabajar en modos distintos. Estos tipos a los que hacemos referencia son los siguientes:

1. **Sistemas operativos "batch" o secuenciales por lotes.** Permiten ejecutar los trabajos uno a uno de una manera estrictamente secuencial. Los programas se pueden ejecutar nada más ser introducidos, o memorizarse en algún dispositivo de almacenamiento para ejecutarse más tarde, Un sistema funcionando en modo "batch" no necesita la atención del usuario durante la ejecución de los trabajos, en contraposición con los tipos siguientes, que son sistemas interactivos con el usuario.





Ejecución de trabajos por lotes.

2. **Sistemas operativos monousuario.** Sólo son capaces de atender exclusivamente a un usuario (de ahí su nombre). Son el tipo de sistemas operativos que trabajan sobre la mayoría de los ordenadores personales. Tienen la ventaja de que su aprendizaje y utilización es muy sencilla, pero, por contra, sus prestaciones son más bien escasas.

3. **Sistemas operativos multitarea.** Con ellos el ordenador es capaz de ejecutar varias tareas (varios programas) de manera simultánea, lo cual no implica la existencia de varios procesadores. Estas tareas pueden ser encomendadas por un mismo usuario, el cual puede supervisarlas a la vez mediante las oportunas ventanas abiertas en pantalla.

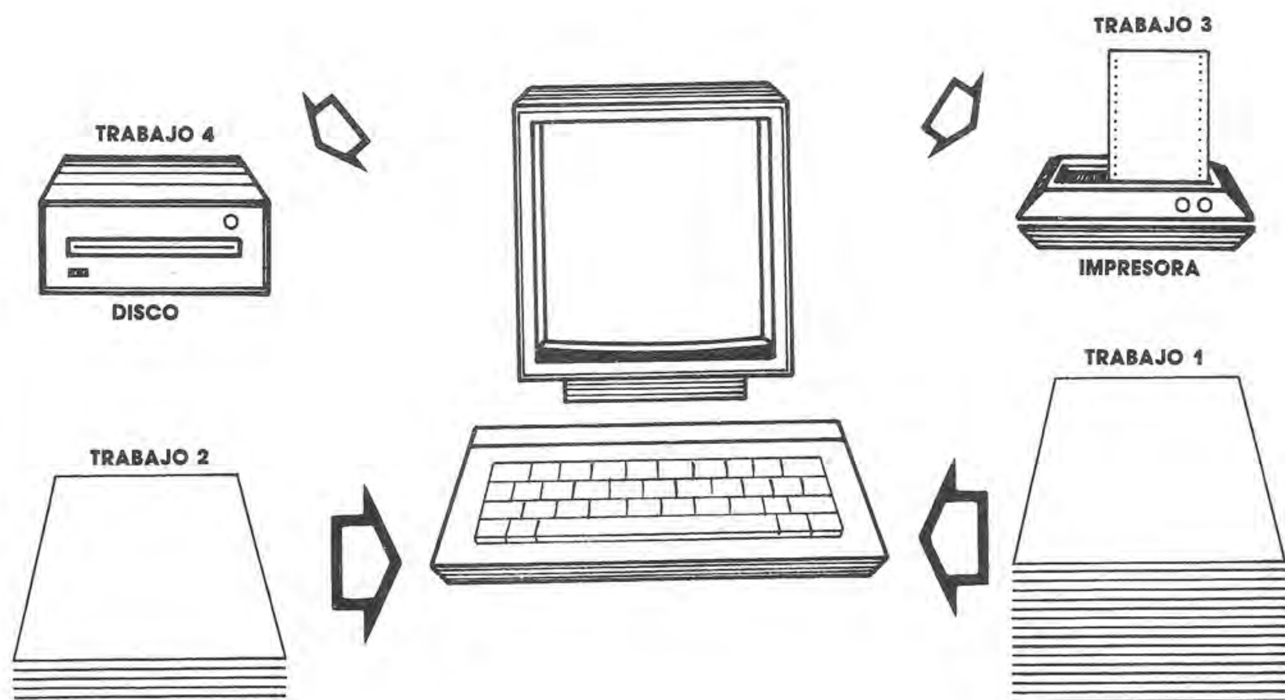
4. **Sistemas operativos multiusuario.** Permiten que el ordenador reparta su atención entre varios usuarios o puestos de trabajo, ejecutando en cada caso una sola tarea distinta (en caso de que el sistema sólo tenga un procesador central). No obstante, la velocidad de los procesadores es tal que los tiempos de espera de los usuarios son casi insignificantes.

5. **Sistemas operativos en tiempo real.** El ordenador debe ser capaz de recibir datos tan rápidamente como lleguen, ya que en caso contrario se perderán.



Sistemas operativos para ordenadores personales

Hasta la aparición del primer microprocesador de 8 bits (el Intel 8080), capaz de configurar un microordenador tal y como hoy en día los conocemos, el concepto de sistema operativo estaba encuadrado en el ámbito de los miniordenadores y grandes sistemas (mainframes). Fue necesaria la aparición del microprocesador Z-80 (de Zilog) para que se produjera la primera implantación generalizada de un sistema operativo para microordenadores, que vino de la mano del CP/M, siglas de Control Program for Microprocessors (Programa de Control para Microprocesadores). Durante algunos años el CP/M fue el sistema operativo casi exclusivo de los microordenado-





Ejemplo de cómo trabaja un ordenador con S.O. multitarea.

res, ya que al adoptar este sistema los fabricantes permitían a sus equipos acceder a toda la variada gama de programas adaptados a CP/M que se estaba creando. Su cada vez mayor aceptación revirtió en el nacimiento de una biblioteca (cada vez más voluminosa) de aplicaciones que se podían ejecutar en cualquier máquina, siempre que ésta trabajara bajo CP/M. Fue así como se convirtió en el sistema operativo por excelencia de los 8 bits.

Otro sistema para micros de 8 bits (concretamente para el Rockwell 6502) es el APPLE.DOS. Este quizá sea, después de CP/M, el más implantado, debido sobre todo a que la firma norteamericana Apple fue una de las pioneras de la revolución microinformática. A pesar de la exclusividad de este sistema operativo (sólo se encuentra en microordenadores Apple y compatibles de otras firmas), su nivel de difusión es alto debido a la gran cantidad de programas de aplicación desarrollados para él en los últimos años.

Importantes, también en el entorno de los 8 bits, son los sistemas operativos UCSD p-SYSTEM, desarrollado en la Universidad de San Diego, en California, en un lenguaje de programación de alto nivel como es el Pascal, y OASIS, primer sistema operativo multiusuario creado para el microprocesador Z-80.

Con la aparición del microprocesador Intel 8086, primer miembro de la familia Intel de 16 bits, comienza la historia de un nuevo sistema operativo, el MS/DOS, que actualmente es un estándar (en el ámbito de los micros de 16 bits) gracias a que IBM lo adoptó en 1981 como sistema operativo para su ordenador personal (el famoso IMB PC).

ORDENADOR CENTRAL

TERMINAL 1

TERMINAL 2

TERMINAL 3

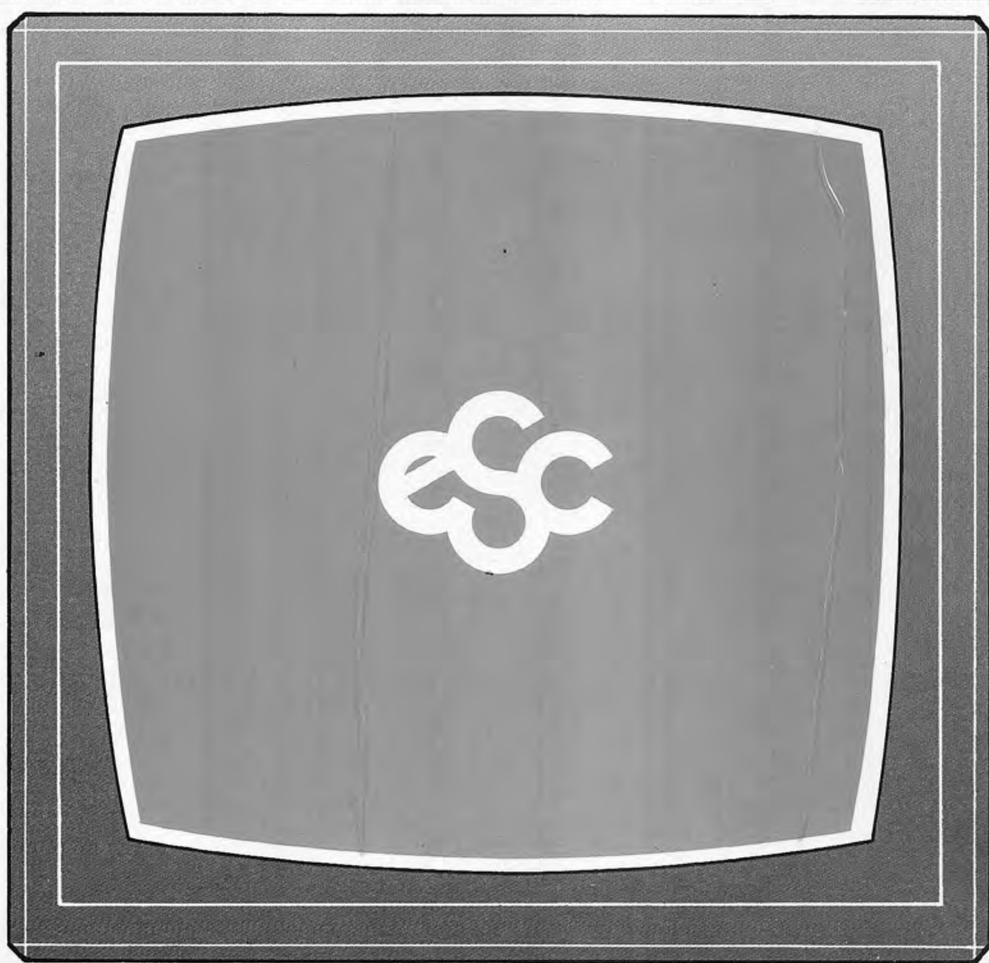


Un ordenador central controla varios terminales mediante un sistema operativo multiusuario.

Por último, un sistema que últimamente se está implantando con fuerza, sobre todo en los ordenadores personales de mayor potencia, es UNIX, aunque ya nos movemos en un entorno multiusuario, a diferencia de los sistemas operativos anteriormente comentados que son monousuario (si bien, existen versiones multipuesto para alguno de ellos, por ejemplo, CP/M).

La concepción de UNIX es totalmente distinta a la de MS/DOS o CP/M, ya que estos dos últimos fueron concebidos para su uso exclusivo en microordenadores, mientras que UNIX fue concebido para su uso en grandes sistemas. Los avances habidos en tecnología electrónica y la mayor potencia cada vez de los microprocesadores, han hecho posible su migración hacia sistemas microinformáticos, con lo cual hoy en día es posible encontrar UNIX en micros, minis y grandes ordenadores.


Ambito de los sistemas operativos para ordenadores personales.



▼ EDICIONES ▼ SIGLO ▼ CULTURAL ▼